International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Optimizing Adversarial Attacks on Graph Neural Networks via Honey Badger Energy Valley Optimization

Mr. Ganesh Ingle *[1], Dr. Sanjesh Pawale [2]

Submitted: 27/01/2024 **Revised**: 05/03/2024 **Accepted**: 13/03/2024

Abstract: In recent years, Graph Neural Networks (GNN) has gained considerable attention due to the practical importance of graph structure data in graph representation learning. It is most commonly utilized in fraud detection, privacy-inference attacks, completion of knowledge graphs, item recommendation, and so on. The GNN is highly vulnerable to adversarial attacks, which affect the reliability of the system, reduce the accuracy of prediction on test data, and increase the loss function of training data. However, the existing approaches utilized to reduce the impacts of adversarial attacks on GNN focus only on highly linked training process. Thus, a GNN_Attacker model is designed in this research for the generation of adversarial attacks in GNN. The binary image is allowed for graph construction and the adversarial attacks are generated in the constructed graph using GNN. Here, the Energy Honey Badger Optimization (EHBO) is introduced for the generation of training samples and GNN is again utilized for testing the generated adversarial attacks. Moreover, the adversarial attack generation performance of GNN_Attacker is validated. It demonstrates that the GNN_Attacker attained superior performance with maximum visual similarity, classification accuracy, and attack success rate of 90.77%, 94.68%, and 96.54% respectively.

Keywords: *Energy valley optimization, Honey Badger Optimization Algorithm, Graph Neural Network, Energy Honey Badger Optimization*

1. Introduction

Graphs are considered as a data structure used for general purposes that comprises entities represented by edges and nodes. In recent years, graph-based machine learning techniques have been widely utilized in different applications, like graph classification, community detection, link prediction, and semi-supervised learning [19][8]. The relation among the complex systems and entities is characterized by the effective representation of graph structure. The real-world complex applications of graph theory in technical, semantic, biological, and social networks have been analysed by many researchers based on network theory over the past several decades. The area of graph data mining has gained huge attention among researchers due to the generation of huge volumes of graph data via graph representation and real-world networked systems of independent samples. Moreover, more efforts are given to focus on link prediction utilized for the detection of spurious links as well as uncovering missing links by using intrinsic structural features of the graphs. The performance of knowledge-related tasks is increased by using link prediction in various applications [20][3]. The increasing growth of graph-based machine learning techniques, such as graph-based approaches and other machine learning approaches makes the models vulnerable to adversarial attacks [18][8]. It is necessary to simulate adversarial attacks by learning and assessing the vulnerability and stability of the models in certain cases of deploying the models in a social network environment. The main goal of generating adversarial attacks in graphs is to perform various learning tasks [8].

The modified node embeddings are highly influenced by the adversarial perturbations on raw local data, which creates threats to the server model by making false decisions [2]. GNN and other related modifications are becoming mainstream approaches that gained significant attention among researchers. The GNN is used to understand the representation of graphs in various fields, like graph generation, combinatorial optimization, knowledge mapping, natural science research, computer vision, and natural language processing [17][1]. The wide application of GNN encountered different security issues, where incorrect predictions may occur due to imposed slight deliberate perturbations in the GNN model [24][25]. Advanced learning models like graph convolution are generally utilized by GNN as compared with other conventional models for the extraction of latent information from neighbours of node and to record high performance while performing downstream tasks that includes node classification. The GNN's deep learning nature makes them highly vulnerable to the generation of adversarial attacks. The GNN is applied to security-critical tasks due to the potential increase of security concerns of the model and their reasoning capability as well as powerful learning ability. Generally, adversarial

[1] *Department of Computer Engineering, Vishwakarma University, Pune*
ORCID ID: 0000-0001-5628-0388
[2] *Department of Computer Engineering, Vishwakarma University, Pune*
ORCID ID: 0009-0006-7556-2528
** Corresponding Author Email: ganesh.ingle-308@vupune.ac.in*

attacks on graph data are categorized into two distinct types, such as structural perturbation-based techniques and attribute perturbation-based techniques. The attribute perturbation-based attacks techniques misclassify the GNN by perturbing the attributes of edges or nodes to follow the adversarial attacks on images [16][4].

The relational knowledge of nodes is highly utilized by graph-based methods to effectively change the structure of a graph while performing graph tasks by attacking conventional graph-based methods [15][4]. The reasoning or training efforts of GNN are highly deteriorated due to changes that occur in the network topology. Thus, the paradigm of structural perturbation-based attacks is prompted by a large number of studies [14]. Moreover, the perturbations are adopted to increase the robustness of the model during the process of adversarial training. The graph embeddings are effectively performed by using the GNN-based adversarial training approaches. Similarly, a robust graph representation learning approach is designed directly to effectively generate adversarial attacks [26]. The Regional Graph Convolutional Neural Network (R-GCN) is used for the creation of adversarial attacks directly, which also effectively reduces the various negative impacts that occur due to adversarial attacks [27]. In addition, various databases are utilized to perform adversarial training to obtain highly robust models as compared with the existing models [5]. The majority of traditional adversarial attack generation approaches utilize query limitations while generating graph-based adversarial attacks. The graph classification carried out by executing adversarial attacks generally train the attacking agents by accessing a portion of the test set, require high computational time, and requires to query the target model [8]. Hybrid optimization algorithms are also employed to fine-tune the models while providing solutions to various adversarial attacks on graph classification.

This research presents a GNN_Attacker model for adversarial attack generation in GNN. The graph is constructed from the input binary image taken from the database initially. Afterward, the generation of adversarial attacks is performed on the constructed graph by using GNN. Moreover, the newly designed algorithm approach, EHBO is used for the generation of training samples by considering fitness functions, such as visual similarity and classification accuracy. The generated training samples are utilized to fine-tune the GNN model. Also, the generated adversarial attacks are tested again by using GNN.

The main research contribution is,

- ***Designed GNN_Attacker for adversarial attack generation****:* The GNN_Attacker is designed for the generation of adversarial attacks in GNN. The training of GNN is performed by using the EHBO approach based on fitness functions, like visual similarity and classification accuracy. The EHBO is designed by integrating Energy Valley Optimization (EVO) and Honey Badger Optimization Algorithm (HBA) approaches.

The article is arranged as, the analysis of traditional attack generation approaches is elucidated in section 2, and section 3 explicates the adversarial attack generation performance of GNN_Attacker. Moreover, section 4 portrays the validation of experimental outcomes with discussions and the article is concluded in section 5.

2. Motivation

The GNN has gained huge attention in various research areas, like traffic, social networking, and e-commerce that performs link prediction, node classification, and so on with high performance. As compared with other networks, the GNN is highly susceptible to adversarial attacks that severely influence the practical application of GNN. The prevailing techniques used for the generation of adversarial attacks in GNN were not successful in updating the graph which also affects the robustness of the model. Hence, a GNN_Attacker is introduced for the adversarial attack generation in GNN.

2.1. Literature Survey

Wu, Y., *et al*. [1] designed Parattack model Training for the creation of adversarial attacks against graph data. This model significantly detected model training parameters before and after the generation of the attack. It effectively verified the hypothesis of parameter discrepancy rationality and achieved high accuracy, but it failed to differentiate various attack directions by analyzing the parameters. Chen, J., *et al*. [2] developed Graph-Fraudster for the creation of adversarial attacks by embedding noise-added global nodes. It generated attacks by stealing the embeddings of the global node and creating the server's shallow model. This technique recorded very low network complexity, but it encountered severe leakage of information while generating adversarial attacks in GNN. Xian, X., *et al*. [3] established Deep Ensemble Coding (DeepEC) for the creation of adversarial examples by utilizing a structure enhancement mechanism. In this model, an evolutionary perturbation-based link selection mechanism was presented to demonstrate the performance of adversarial attack generation. It recorded very low execution time during the generation of adversarial attacks. However, due to the absence of structure estimation of input data malicious tasks were also generated in this model. Zhang, C., *et a*l. [4] introduced a Saturation adversarial Attack with Meta-gradient (SAM) to obtain the necessary information on the gradient by considering structural perturbations of the graph. The attack efficiency of the model was effectively improved by flipping multiple edges of the determined meta-

gradients in one query. This model determined query-based poisoning attacks and possessed high pooling capacity during the generation of global attacks against graph node classification. However, it was not successful in analyzing the stealthiness of adversarial attacks in GNN.

Qiao, Z., *et al*. [5] designed a Graph transformation mechanism for the transformation of the graph. This approach was highly suitable for the generation of adversarial attacks under four transformation forms to optimize the defense flow against the attacks on graph data. It effectively covered the defense effectiveness of the model under less computational time. However, because of excessive transformation, the model recorded high damage to the harmless structures. Wu, X.G., *et al*. [28] developed Calibrated Co-training (C^2oG) to identify the effects of adversarial perturbations. The structure information as well as feature information was effectively integrated to design the C2oG model. This model was simple to implement and effectively increased its robustness against adversarial attacks, but the model suffered from high computational complexity issues during the defense against adversarial attacks. Alarab, I. and Prakoonwit, S., [7] introduced Monte-Carlo based on Adversarial Attacks (MC-AA) to capture the uncertainty of the model by using the adversarial attack idea. It effectively classified binary nodes by estimating the uncertainties and examined the viability of the model. This model significantly captured the wrong labels in overlapping regions, but it failed in multiclass node classification. Wan, X., *et al*. [8] established a Bayesian optimization-based attack method for the classification of graph models. It linked the vulnerability of graph-based machine learning models by analyzing the generated adversarial examples. It effectively solved the vulnerabilities of graph classification systems that largely outweigh the risks. However, it failed to utilize various mainstream victim models to verify the robustness of the model while generating adversarial examples. Ganesh Ingle, *et.al.* [6,12,13] explores the optimal masking ratio presents a significant research gap, as understanding the precise balance between model accuracy and robustness against adversarial attacks remains unresolved. Further investigation into detailed masking ratios and the creation of adaptive masking strategies, which adjust according to a model's exposure to adversarial threats, is crucial. Additionally, the effectiveness of novel adversarial attack strategies highlights the potential shortcomings of existing Graph Neural Network (GNN) defense mechanisms. There's a clear necessity for the development of adaptive defense strategies that can dynamically evolve in response to the continually changing nature of adversarial attacks, particularly leveraging internal model insights like Class Activation Mapping (CAM). Moreover, the successful application of Input Adversarial Training (IAT) in controlled experiments prompts the need for research into its deployment in real-world power quality management systems. The scalability, efficiency, and real-time processing capabilities of IAT in environments with highly variable data and operational conditions represent pivotal areas for future exploration. These gaps underscore the ongoing need for advancements in defensive techniques to secure neural network models against sophisticated adversarial threats. Ingle, G.B., *et.al.* [21] delve into the practical implications of adversarial attacks and defenses across various applications, from image recognition and autonomous vehicles to cybersecurity and fraud detection. These discussions highlight the real-world significance of improving model resilience, emphasizing the need for ongoing research and development in adversarial machine learning.

2.2. Challenges

The limitations encountered by classical adversarial attack detection techniques are demonstrated below,

- The Parattack model used in [1] effectively resisted adversarial attacks presented in various real-time applications, but the model only recorded fuzzy attack directions and deviations from the correct attack direction due to the optimization of different parameters.
- The Graph-Fraudster developed in [2] was highly robust in the generation of attacks. However, this model failed to consider node features for perturbation and was not successful to reduce the impact of data imbalance of attacks.
- The DeepEC used in [3] achieved high computational efficiency during adversarial attack generation. However, the generalization of link prediction models was affected because the corrupted version of inputs and the adversarial examples generated by different attacks were not incorporated into the design process.
- The SAM model employed in [4] increased the performance of iterative attacks more significantly, but it failed to quantify and utilize the black-box setting to learn the vulnerability of GNN to adversarial attacks.
- The traditional adversarial attack generation techniques extracted latent information from neighboring nodes and achieved good performance on various downstream tasks. However, it failed to apply attacks to the attributed graphs that involve features associated with edges or nodes. It also focused only on determining the node classification performance of the model and posed severe impacts on the security of the model

3. Designed GNN_Attacker for the Generation of Adversarial Attacks

In this manuscript, a GNN_Attacker is developed for the generation of adversarial attacks in GNN. The binary image acquired from the database is initially allowed for the construction of images into graphs. Then, the adversarial attacks are generated in graphs by using GNN [9], where the training samples are generated using EHBO by considering fitness functions, like similarity and classification accuracy. Here, the EHBO is generated by the incorporation of EVO [11] and HBA [10]. In addition, the testing of generated adversarial attacks is performed by using GNN. Figure 1 displays the block diagram of the designed GNN_Attacker for the generation of adversarial attacks in GNN.

3.1 Image Acquisition

The input image taken for the generation of adversarial attacks from the database is initially given by the expression designated as,

$$A = \left[A_1, A_2, A_3, .., A_I, ..., A_R \right] \tag{1}$$

here, the total number of images available in the database is symbolized as R, the database considered for the generation of adversarial attacks is represented as A, and the I^{th} image taken for adversarial attack generation is signified as A_I.

3.2 Construction of Image to Graph

A graph is a non-linear structure that comprises edges and vertices, where the vertices are termed nodes and edges are lines that interconnect two nodes. The image-to-graph construction is performed by initially allowing the two-dimensional (2D) input image A_I into the form of the matrix. The pixel values "0" and "1" are provided to each cell of the matrix and its corresponding nodes are plotted. The edges are constructed, if the value of one node is the same as the other, thus constructing the corresponding graph of the input image A_I. For example, the conversion of the input image A_I into matrix format is displayed in figure 2, and pixel values "0" and "1" are given to each cell of the matrix.

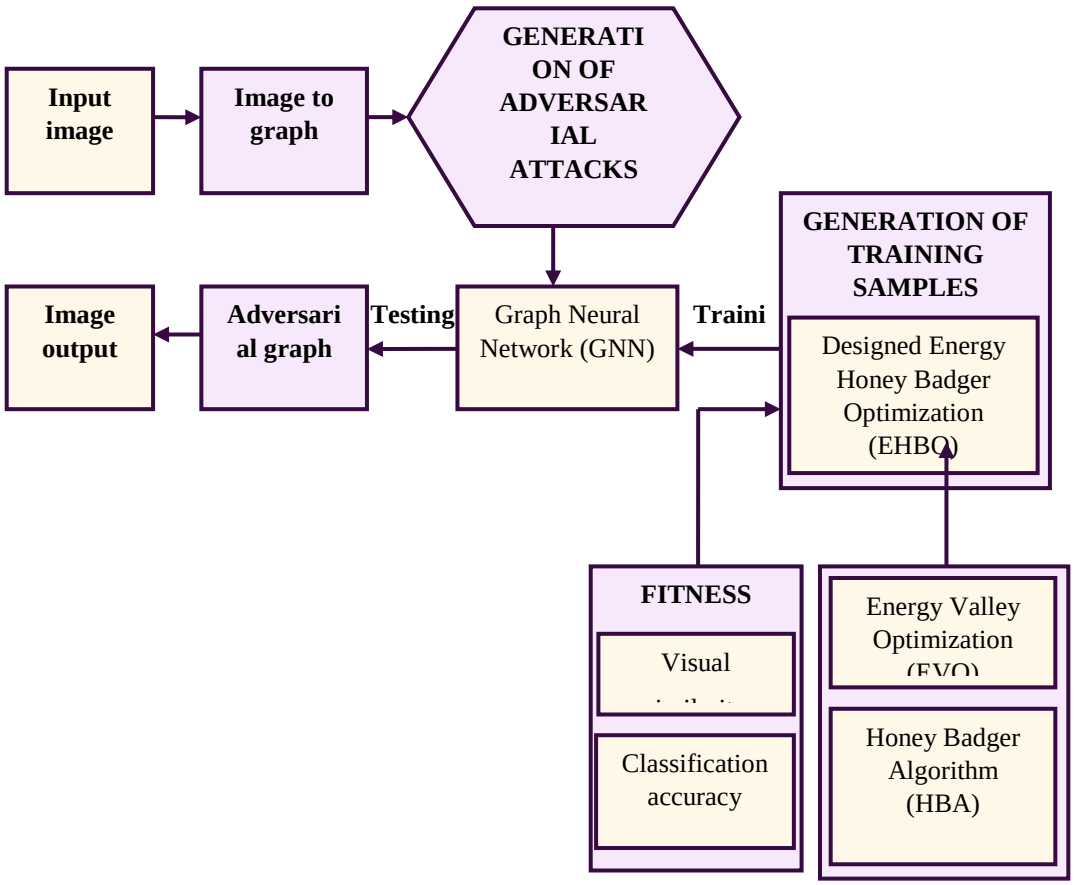


Fig. 1. Block diagram of designed GNN_Attacker for the generation of adversarial attacks in GNN.

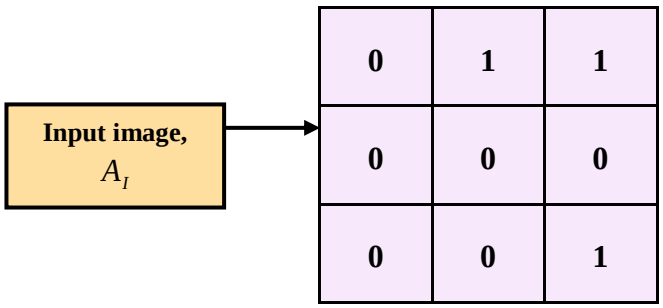


Fig. 2. Matrix representation of input image

Moreover, the node $N_a; [a = 1,2,3,...,J]$ is assigned to each cell of the given matrix and is displayed in figure 3.

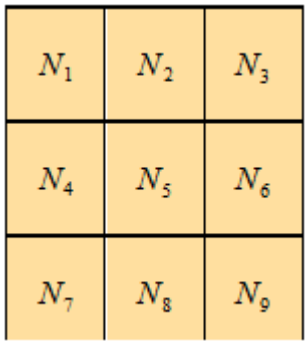


Fig 3. Assignment of node to each cell

Finally, the graph is constructed by creating connections between the nodes by using edges. If the adjacent pixels have a non-zero value, then an edge is considered to exist between them. For example, the pixel value of N_2 "1" is the same as the pixel value of N_3 "1" in the matrix i.e. $N_2 = N_3$, then the edge is constructed between the node N_2 and N_3 and is given by $N_2 N_3 = 1$. Moreover, if the pixel values are not equal to unity, then no edge exist between them. For example, the pixel value of N_1 is "0" and the pixel value of N_2 is "1", then no edge exist between them i.e. $N_1 \neq N_2$, which is given by $N_1 N_2 = 0$. Similarly, the edge is constructed for all nodes with pixel values. The graph construction performed between nodes is displayed in figure 4 and the graph constructed is further fed into GNN for the generation of adversarial attacks.

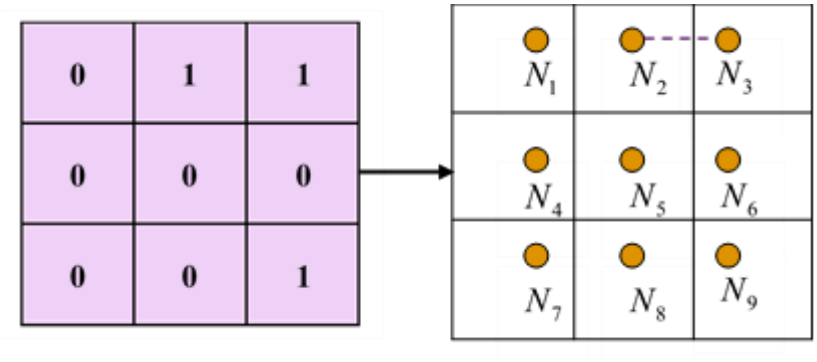


Fig 4. Construction of graph from the input image

3.3 GNN Architecture

The GNN model [9] is the extension of random walk models as well as recursive neural networks. The GNN can effectively process different general class of graphs that involves undirected, directed, and cyclic graphs it becomes an extension of recursive neural networks, and also effectively handle the node-focused application. In general, GNN is based on an information diffusion

mechanism, where a set of units is used to process the graph and each unit corresponds to the node of the graph that is linked based on graph connectivity. Until reaching a stable equilibrium, the units constantly modify their state and transfer information. At each node, the output of GNN is estimated by considering the unit state.

In GNN, the graph node indicates the concepts or objects, and the relationship signifies the edges of the graph, where the related concepts and their features are used to define each concept. Thus, a state $a_i \in R^m$ is attached to each node i by considering the information available in the neighbourhood i. The output G_i is obtained to decide on the concept by representing the concept i of state a_i. Let us consider a local transition function R_c and local output function Q_c for describing the generation of output. Then, the expression of a_i and G_i is given by,

$$a_i = R_c\left[L_i, L_S, a_E, L_N\right] \tag{2}$$

$$G_i = Q_c\left[a_i, L_i\right] \tag{3}$$

where, the labels of states, edges, and the labels of i is signified as $L_S, a_E,$ and L_i, the label of nodes in the neighbourhood i is represented as L_N.

Let us consider the vectors $a, G, L,$ that L_N are generated by stacking the states, outputs, labels, and labels of nodes. Thus, the equation (2) and (3) can be rewritten using the expression given by,

$$a = r_c\left[a, L\right] \tag{4}$$

$$G_i = q_c\left[a, L_N\right] \tag{5}$$

here, the stacked version of $|N|$ instances of R_c and Q_c for the global transition function is signified as r_c and the global output function is denoted as q_c. The expressions (4) and (5) are used to represent a map that utilizes graph input and provides output G_i for each node. Thus, Banach fixed point theorem [29] provides unique solutions for the equations by considering R_c as a contraction map concerning state, i.e. $\beta, 0 \le \beta < 1$. It is given by

$\|R_c(a,L) - R_c(b,L)\| \le \beta\|a-b\|$ for any a, b. Here, the vertical norm is represented as $\|.\|$. Therefore, the contraction map R_c is considered for the moment in GNN and the transition function is implemented appropriately to enforce this property.

The equations (2) and (3) are used to process both positional as well as non-positional graphs. The position of the neighbours is received by R_c as extra inputs in positional graphs, which can quickly record the information available in $a_E, L_E,$ and L_N sort based on the position of neighbours. Moreover, the information is padded properly using the special position of null values corresponding to nonexistent neighbours. For example, $a_E = \left[b_1, b_2, b_3, ..., b_F\right]$, here F indicates the maximum number of neighbours in the node. If m is the neighbour of n^{th} neighbour of i is $b_n = a_m$ and if the n^{th} neighbour is not available i.e. for the pre-defined null state a_0 is $b_n = a_0$. Moreover, the function R_c of equation (2) can be replaced for a nonpositional graph by using the expression,

$$a_i = \sum_{m \in N} g_c\left[L_i, L_{(i,m)}, a_m, L_m\right] \tag{6}$$

here, the parametric function is represented as g_c and the nonpositional form of the graph is referred in equation (2) and equation (3), where the positional form is given in equation (6). Thus, the graph output O_I with the generated adversarial attack is obtained from GNN when the constructed graph is fed into the GNN structure. Thus, the resultant constructed graph output generated by GNN with adversarial attacks is elucidated in figure 5.

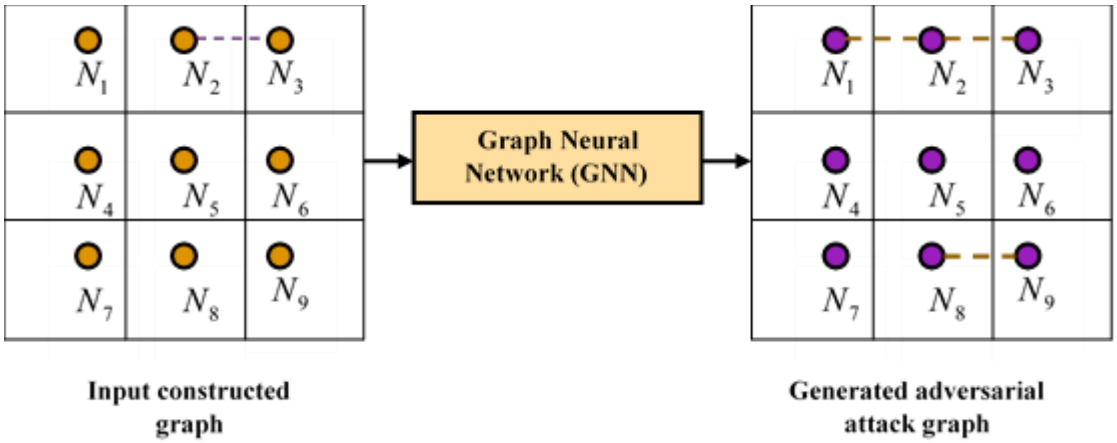


Fig 5. Generation of adversarial attacks using GNN

The adversarial output image is finally obtained from the resultant generated adversarial attack graph by giving pixel values "0" and "1" to each cell of the matrix. The pixel value of "1" is provided for each node with constructed edges and the pixel value of "0" is given to nodes, if the edges are not constructed. Thus, the adversarial output image I_A is obtained and figure 6 depicts the adversarial output image obtained from the generated adversarial attack graph.

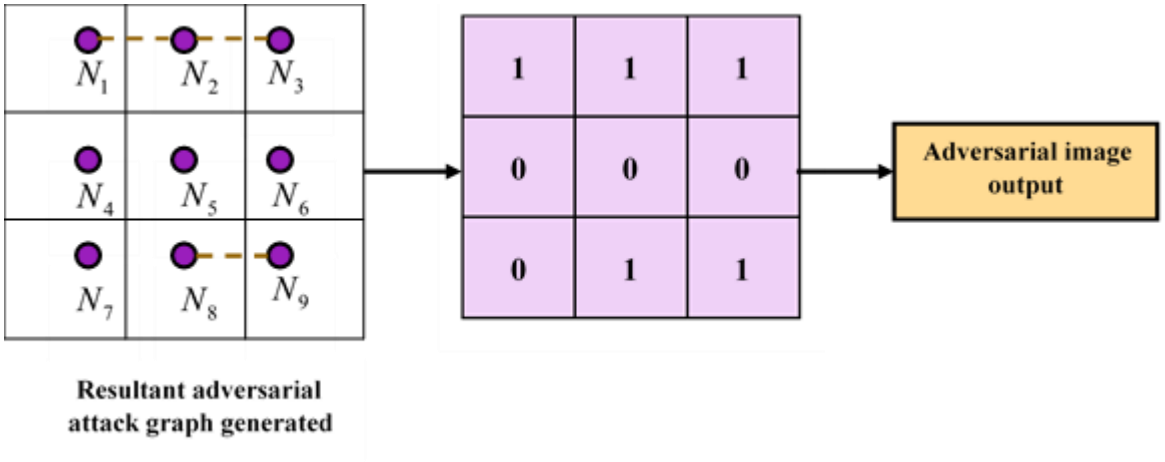


Fig 6. Generation of adversarial image

Generation of Training Samples

The training samples are generated by using the EHBO algorithm together by encoding solution and determining fitness function, where the process performed is enumerated below,

Solution Encoding

The solution encoding is performed by selecting optimal solutions randomly to accomplish the task. The optimal solution is determined by encoding training samples along the edges for different nodes N_1 to N_9, where the representation of solution encoding is given in figure 7.

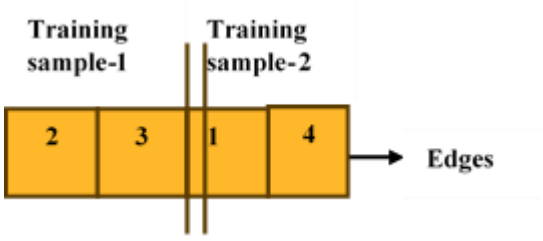


Fig 7. Encoding of training samples

Fitness Estimation

The fitness function is estimated using the expression formulated as,

$$Fitness\ function = \sum_{S=1}^{I}\left[\frac{\eta+\mu}{2}\right] \quad (7)$$

here, the total training is represented as I, η indicates visual similarity, and μ denotes classification accuracy.

-Visual similarity (η): It is defined as the similarity among the visual characteristics of input and output images.

-Classification accuracy (μ): The classification accuracy is used for the determination of corrected edges, where it is the relationship among the original and targeted image output determined by GNN. The classification accuracy is formulated as,

$$\mu = \frac{\psi_1 + \psi_2}{\psi_1 + \psi_2 + \psi_3 + \psi_4} \tag{8}$$

where, the term ψ_1 and ψ_2 represents true positive and true negative, ψ_3 and ψ_4 denotes false positive and false negative.

3.4 Designed EHBO Model for the Generation of Training Samples

The training samples used for the training of GNN_Attacker are generated by using EHBO algorithmic technique. The algorithmic approaches EVO [11] and HBA [10] are integrated to form the design EHBO approach. The advanced principles of physics under different modes as well as particle decay stability are considered for designing EVO. In general, the EVO converges quickly and it helps to identify the solution with the lowest possible objective function. Similarly, HBA is designed by considering the different inspecting characteristics of honey badger animals during the location of food. The honey badger is a mammal with black and white furs that are commonly found in tropical rainforests of Africa and is well capable of climbing trees for food. The honey badger uses digging and honey phases for catching its prey and to locate beehive, whereas in the honey phase it follows the honeyguide birds to reach the beehive. Moreover, in the digging phase, it performs actions that are similar to the Cardioid shape. The EVO acts as an approximation algorithm and was not successful in providing accurate solutions. Hence, the HBA is incorporated with EVO for the determination of the exact solution, and the mathematical modelling of EHBO is designated as follows,

Step 1: Initialization: The solution is randomly initialized in the search space and is designated as,

$$I = \{I_1, I_2, \ldots, I_V, \ldots, I_Y\} \tag{9}$$

here, the total number of solutions presented in the search space is indicated as Y, and I_V signifies the V^{th} candidate in the random search space.

Step 2: Estimation of fitness: The optimal solution is determined using the expression given in equation (7).

Step 3: Stability of search space: In the search space, the different stability level particles are considered during the initialization of the candidate solution and are designated as,

$$H_u^g = H_{u,\min}^g + R.\left(H_{u,\max}^g - H_{u,\min}^g\right), \quad \begin{cases} u = 1,2,\ldots,Y \\ g = 1,2,\ldots,X \end{cases} \tag{10}$$

Where,the problem dimension is signified as X, Y indicates total particles in the universe, H_u^g denotes the determination of the initial position of u^{th} iteration of g^{th} dimensional variable, $H_{u,\min}^g$ and $H_{u,\max}^g$ symbolizes lower and upper bound, and R indicates the random number that is set to range [0,1].

Step 4: Enrichment of particle: The particle enrichment bound is determined by considering the difference between neutron-poor and neutron-rich particles. The α rays are emitted to increase the stability of products by considering physical principles. Moreover, the best level of stability is substituted in rays to remove the decision variable of the candidate solution and is expressed as,

$$H_u^N = H_u\left[H_{BS}\left(H_u^g\right)\right], \quad \begin{cases} u = 1,2,\ldots,Y \\ g = Alphaindex\ II. \end{cases} \tag{11}$$

here, H_u^N represents the particles generated newly, H_u denotes the current vector position of the particle, H_{BS} indicates the best stability level of particles, and $Alphaindex\ II$ signifies the emitted α rays. Similarly, the gamma index is considered to enhance the stability level of particles. Thus, the second candidate solution is designated as,

$$H_u^{N2} = H_u\left[H_{NP}\left(H_u^g\right)\right], \quad \begin{cases} u = 1,2,\ldots,Y \\ g = Gammaindex\ II. \end{cases} \tag{12}$$

Where, H_{NP} denotes the position vector of the neighbouring particle.

Step 5: Beta decay: The unstable particles perform beta decay and the particle position is modified, if the particle

level of stability is lower than the stability bound, which is expressed as,

$$H_u^{N1} = H_u + \frac{\left(J_1 \times H_{BS} - J_2 \times H_{PC}\right)}{LS_u}, \ u = 1,2,...,Y \tag{13}$$

Here, H_{BS} signifies the vector position of the particle with best stability level, H_{PC} indicates the vector position of centre of particles, the stability level is represented as LS_u, and J_1 and J_2 are random numbers fixed at [0,1].

Step 6: Exploitation and exploration level: The position of the algorithm's exploration and exploitation levels are increased by modifying the particle's position using beta decay and is given by,

$$H_u^{N2} = H_u + \left(J_3 \times H_{BS} - J_4 \times H_{NP}\right), \quad u = 1,2,...,Y \tag{14}$$

where, the random numbers are signified as J_3 and J_4, which is fixed to [0,1].

Let us consider, $H_u^{N2} = H_u(g+1)$ and $H_u = H_u(g)$. Thus, the equation (14) becomes,

$$H_u(g+1) = H_u(g) + \left(J_3 \times H_{BS} - J_4 \times H_{NP}\right) \tag{15}$$

The HBA [11] is incorporated with EVO [10] for the determination of the exact solution from the search space. From HBA,

$$W_N = W_P + U \times f_7 \times K \times D_I \tag{16}$$

here, the updated position of the honey badger is indicated as W_N, the prey location is represented as W_P, the direction of search space is altered using flag signified as U, and the information of search distance is denoted as D_I. Also, at varying times the influenced search behaviour is given as K, and f is the random number among 0 and 1 which is computed using the expression designated as,

$$D_I = W_P - W_s \tag{17}$$

Moreover, the direction of search space U and search behavior K is determined using the expression given by,

$$U = \begin{cases} 1 & if\ J_6 \le 0.5 \\ -1 & else \end{cases} \quad \text{and} \quad K = P \times \exp\left(\frac{-u}{u_{\max}}\right) \tag{18}$$

here, J_6 denotes random number among 0 and 1, the maximum iteration is signified as $u_{\max}$ and P denotes the constant term which is ≥ 1.

Applying equation (17) in equation (16),

$$W_N = W_P + U \times f_7 \times K \times \left[W_P - W_s\right] \tag{19}$$

Let us consider, $W_N = H_u(g+1)$, $W_s = H_u(g)$, and $W_P = H_P(g)$ and the equation (18) becomes,

$$H_u(g+1) = H_P(g) + U \times f_7 \times K \times \left[H_P(g) - H_u(g)\right] \tag{20}$$

$$H_u(g+1) = H_P(g)\left[1 + U \times f_7 \times K\right] - U \times f_7 \times K \times H_u(g) \tag{21}$$

$$H_u(g) = \frac{H_P(g)\left[1 + U \times f_7 \times K\right] - H_u(g+1)}{\left(U \times f_7 \times K\right)} \tag{22}$$

Substituting equation (21) in equation (15),

$$H_u(g+1) = \left[\frac{H_P(g)\left[1 + U \times f_7 \times K\right] - H_u(g+1)}{\left(U \times f_7 \times K\right)}\right] + \left(J_3 \times H_{BS} - J_4 \times H_{NP}\right) \tag{23}$$

$$H_u(g+1) + \frac{H_u(g+1)}{\left(U \times f_7 \times K\right)} = \frac{H_P(g)\left[1 + U \times f_7 \times K\right] + \left(J_3 \times H_{BS} - J_4 \times H_{NP}\right)\left(U \times f_7 \times K\right)}{\left(U \times f_7 \times K\right)} \tag{24}$$

$$\frac{H_u(g+1)\left(U \times f_7 \times K\right)}{\left(U \times f_7 \times K\right)} = \frac{H_P(g)\left[1 + U \times f_7 \times K\right] + \left(J_3 \times H_{BS} - J_4 \times H_{NP}\right)\left(U \times f_7 \times K\right)}{\left(U \times f_7 \times K\right)} \tag{25}$$

Thus, the updated equation of EHBO is designated as,

$$H_u(g+1) = \frac{H_P(g)\left[1 + U \times f_7 \times K\right] + \left(J_3 \times H_{BS} - J_4 \times H_{NP}\right)\left(U \times f_7 \times K\right)}{\left(U \times f_7 \times K\right)} \tag{26}$$

Step 7: Random movement: A random movement is performed by the particle if it attains a low enrichment level and the particle undergoes positron emission or capture electron to move along the stability band. The executed random movement along the search space is expressed as,

$$H_u^{N2} = H_u + J, \quad u = 1,2,...,Y \tag{27}$$

here, J signifies the random number and is set to [0,1].

Step 8: Check for solution feasibility: The expression of fitness function given in equation (8) is used to compute the optimal solution and if any solution is determined to be more efficient than the present one then the solution can be replaced.

Step 9: Termination: The algorithmic steps given in table 1 are continuously executed until reaching the highest iteration with the optimal solution.

Table 1. Pseudo code of EHBO

SI.No	Pseudo code of EHBO
1	**Input:** Iteration, u
2	**Output:** Optimal best solution H_u
3	***begin***
4	Identify the solution candidate's initial position
6	Determine fitness function by using equation (7)
7	***for*** $u = 1:Y$
8	Evaluate the particle's level of stability
9	Evaluate the particle's neutron enrichment level
10	***if*** $LS_u > BS$
11	Create alpha index I and II
12	***for*** $g = 1: Alpha\ Index\ II$
13	Evaluate using equation (11)
14	***end***
15	Create Gamma Index I and II
16	***for*** $g = 1: Gamma\ Index\ II$
17	Compute using the equation (12)
18	***end***
19	***end if*** $LS_u < BS$
20	Identify the particle center pixel using the expression (13)
21	Identify neighboring pixels using the expression (14)
22	***end***
23	***end***
24	***end while***
25	***Go to the best stability level particle***
26	***end***

Therefore, the suitable samples are generated using the EHBO approach and are used for the training of GNN_Attacker during the generation of adversarial attacks in graphs by taking account of visual similarity and classification accuracy.

4. Results and Discussion

The result obtained by GNN_Attacker used for adversarial attack generation is analyzed and corresponding discussions are performed to determine the generating superiority of the designed model and the analysis carried out is demonstrated below,

4.1. Experimental set-up

The adversarial attacks generation on GNN using the designed GNN_Attacker is implemented using a Python tool.

4.2. Dataset Description

The images considered for the generation of adversarial attacks are taken from the Modified National Institute of Standards and Technology database (MNIST) [22] and the Canadian Institute For Advanced Research (CIFAR) [23] database.

MNIST Database: The database is created mainly to understand the utilization of graph convolutional networks to perform different basic and visual tasks. It is a collection of handwritten digits with 10,000 testing set examples as well as 60,000 training set examples. The database comprises handwritten digits in the form of monochrome images and the digits are centered and size-normalized. The pixel's center of mass is computed and translation is performed to center the images.

CIFAR Database: The CIFAR database comprises about 60,000 images with 10 classes and each image consists of RGB channels of size 32×32 pixels. Among these 60,000 images, there are 10,000 testing images and 50,000 training images. The pixels are vertex in the graph that represents a graph with 1024 vertices.

4.3. Evaluation Parameters

The parameters utilized to determine the generation performance of GNN_Attacker are demonstrated below,

1) Visual similarity: The similarity between the visual characteristics of the input as well as output image is identified using visual similarity.

2) Classification accuracy: It helps to identify the corrected edges of the samples, which is computed using the expression given in equation (8).

3) Attack screen rate: The total adversarial attacks successfully generated by the GNN are computed using the attack success rate.

4.4. Comparative Techniques

The attack generation performance of GNN_Attacker is validated by comparing the performance with existing adversarial attack generation approaches, like the Parattack model [1], Graph-Fraudster [2], DeepEC [3], and SAM [4].

4.5. Comparative Assessment

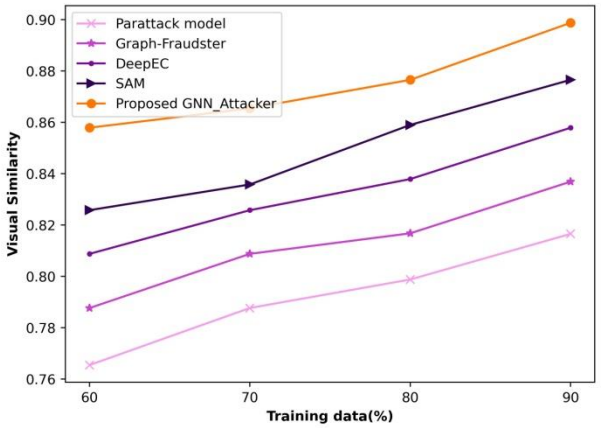
The training data as well as the value of K-fold are varied for the analysis of adversarial attack generation performance of GNN_Attacker using MNIST [22] and CIFAR [23] database, where the analysis performed is demonstrated as follows,

I. MNIST Database

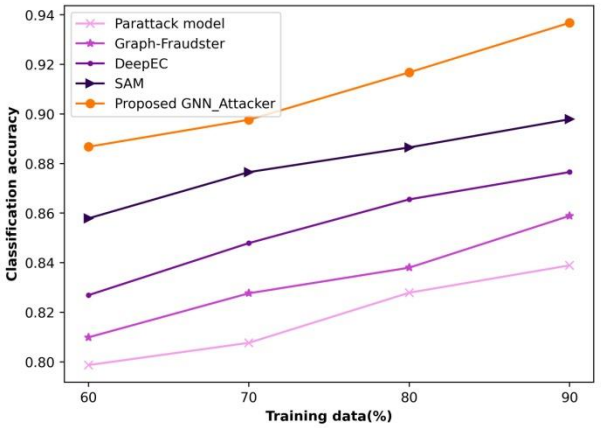
The analysis performed by using the MNIST database utilized for the creation of adversarial attack in the graph is given by

Validation Using Training Data

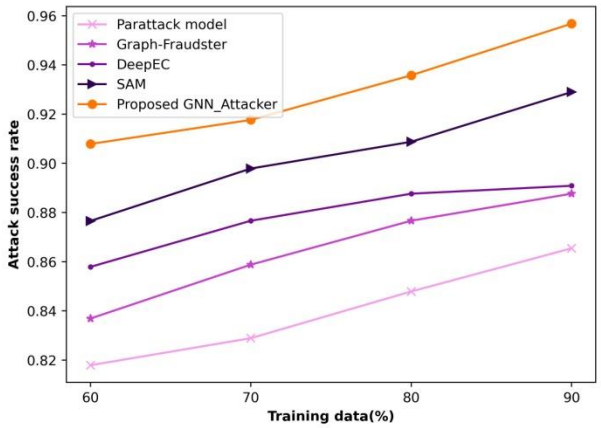
The adversarial attack generation performance of GNN_Attacker using the MNIST database is validated using training data is portrayed in figure 8. The validation of the performance of adversarial attack generation techniques using visual similarity is portrayed in figure 8(a), where the designed GNN_Attacker gained visual similarity of 0.898 for 90% training data. The existing adversarial attack generation techniques, namely the Parattack model, Graph-Fraudster, DeepEC, and SAM obtained visual similarity of 0.816, 0.836, 0.857, and 0.876. The higher performance of 2.47% is attained by GNN_Attacker than the SAM model. The comparative validation by using classification accuracy of different techniques used for the generation of adversarial attacks is depicted in figure 8(b). The classification accuracy of 0.936 is attained by GNN_Attacker for training data of 90%. The classification accuracy recorded by existing approaches is 0.838 by the Parattack model, 0.858 by Graph-Fraudster, 0.876 by DeepEC, and 0.897 by SAM. The GNN_Attacker achieved high performance of 10.45% as compared with the existing Parattack model. Figure 8 (c) shows the analysis of the attack success rate of various adversarial attack generation approaches. The GNN_Attacker gained an attack success rate of 0.956 and the attack success rate recorded by prevailing models, such as the Parattack model is 0.865, Graph-Fraudster is 0.887, DeepEC is 0.890, and SAM is 0.928 for training data of 90%. An improved performance of 7.23% is achieved by GNN_Attacker during adversarial attack generation than the classical Graph-Fraudster approach.



(a)



(b)

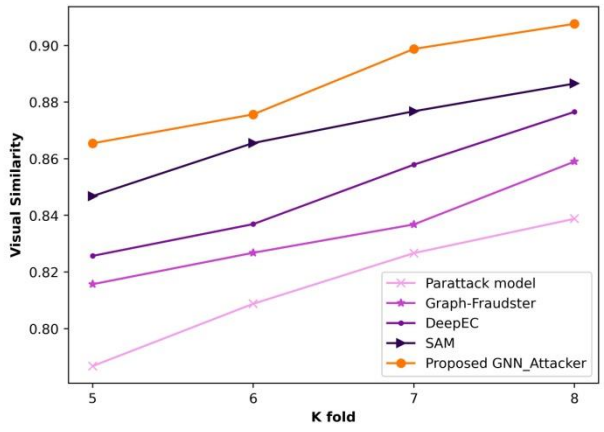


(c)

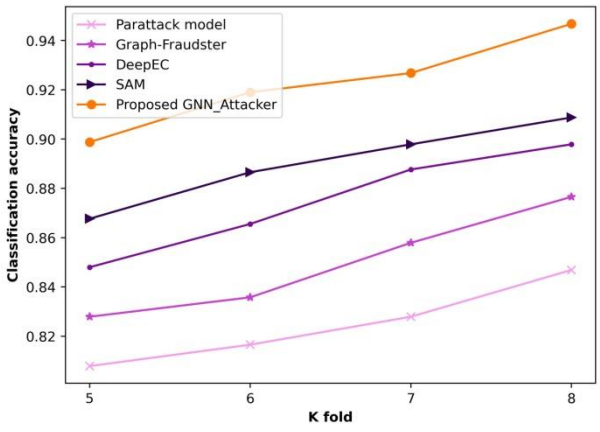
Fig 8. Validation of GNN_Attacker using MNIST database employing training data (a) Visual similarity, (b) Classification accuracy, (c) Attack success rate

Validation using K-fold

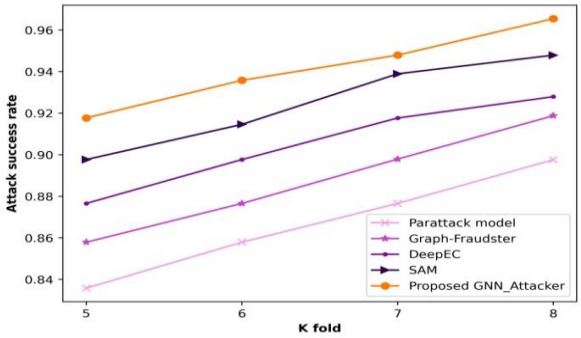
Figure 9 shows the analysis of the performance of GNN_Attacker using the MNIST database employing the K-fold value. Figure 9(a) depicts the validation of the performance of different adversarial attack generation approaches using visual similarity. The GNN_Attacker measured visual similarity of 0.907 for K-fold value 8 and the other existing approaches record visual similarity of 0.838 by Parattack model, 0.858 by Graph-Fraudster, 0.876 by DeepEC, and 0.886 by SAM. The higher performance of 5.36% is attained by GNN_Attacker than the Graph-Fraudster approach. Figure 9(b) portrays the validation of performance using classification accuracy of different techniques used for attack generation. For K-fold value 8, the classification accuracy of 0.946 is recorded by GNN_Attacker, and the classification accuracy of 0.846, 0.876, 0.897, and 0.908 is measured by existing techniques, like the Parattack model, Graph-Fraudster, DeepEC, and SAM. The GNN_Attacker achieved high performance of 7.42% as compared with DeepEC model. The performance analysis of attack generation approaches using attack success rate is depicted in figure 9(c). The attack success rate measured by GNN_Attacker is 0.965 for K-fold value 8, whereas the attack success rate measured by other attack generation techniques, namely Parattack model is 0.897, Graph-Fraudster is 0.918, DeepEC is 0.927, and SAM is 0.947. Here, the maximum performance of 4.83% is attained by GNN_Attacker as compared with the existing Graph-Fraudster model used for attack generation.



(a)



(b)



(c)

Fig 9. Validation of GNN_Attacker using MNIST database employing K-fold value (a) Visual similarity, (b) Classification accuracy, (c) Attack success rate

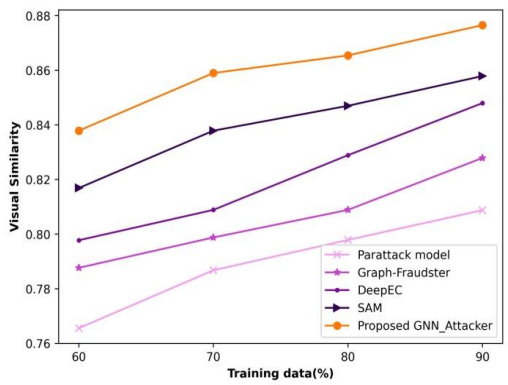
II. CIFAR database

The performance validation of GNN_Attacker using the CIFAR database by using training data and the value of K-fold is enumerated below,

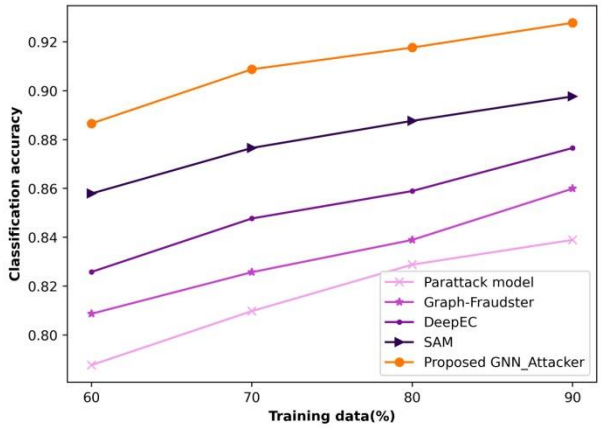
Validation Using Training Data

Figure 10 depicts the performance evaluation of GNN_Attacker using the CIFAR database by varying percentage training data. The performance evaluation of approaches used for attack generation using visual similarity is portrayed in figure 10(a). The visual similarity obtained by GNN_Attacker is 0.876 for training data 90%, where visual similarity recorded by traditional attack generation techniques, namely Parattack model is 0.808, Graph-Fraudster is 0.827, DeepEC is 0.847, and SAM is 0.857. Here, the best performance of 5.55% is achieved by GNN_Attacker as compared with the existing Graph-Fraudster model utilized for attack generation.

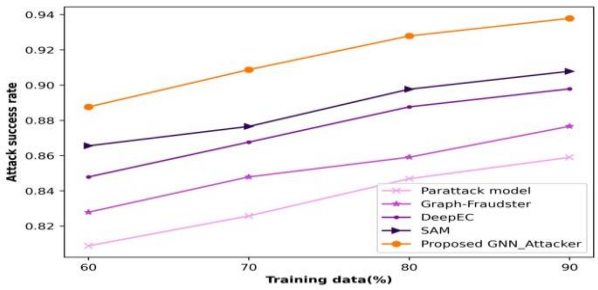
Figure 10(b) elucidates the validation of the performance of various adversarial attack generation approaches using classification accuracy. The GNN_Attacker measured classification accuracy of 0.927 for training data 90% and the other existing approaches record classification accuracy of 0.838 by Parattack model, 0.859 by Graph-Fraudster, 0.876 by DeepEC, and 0.897 by SAM. The higher performance of 7.32% is achieved by GNN_Attacker than the Graph-Fraudster approach. Figure 10(c) illustrates the validation of performance using attack success rate of the different models used for attack generation in graphs. For training data 90%, the attack success rate of 0.937 is recorded by GNN_Attacker, and the attack success rate of 0.858, 0.876, 0.897, and 0.907 is measured by prevailing models, like the Parattack model, Graph-Fraudster, DeepEC, and SAM. The GNN_Attacker achieved high performance of 6.53% than the DeepEC model.



(a)



(b)

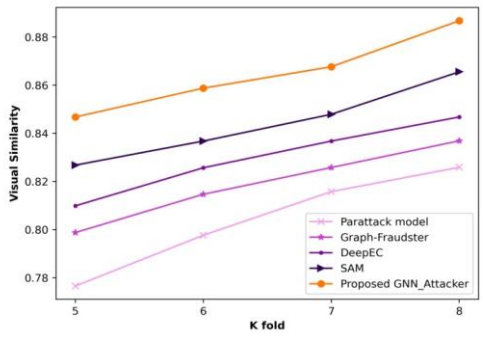


(c)

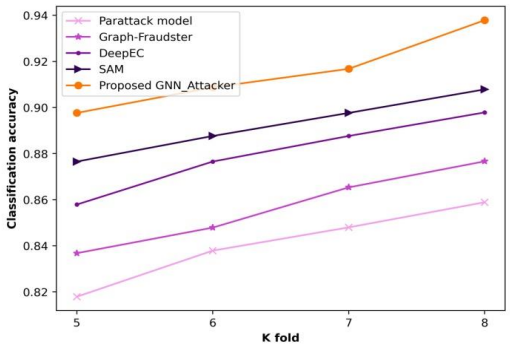
Fig 10. Validation of GNN_Attacker using CIFAR database employing training data (a) Visual similarity, (b) Classification accuracy, (c) Attack success rate

Validation using K-fold

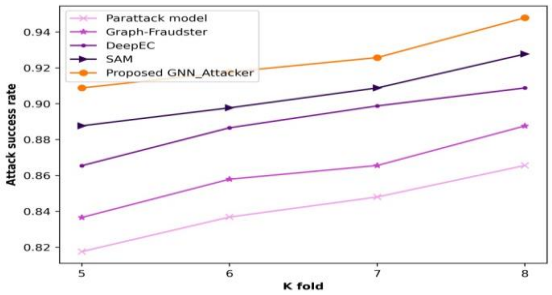
The adversarial attack generation performance of GNN_Attacker utilizing the CIFAR database is analyzed by using the K-fold value is portrayed in figure 11. The comparative evaluation using visual similarity of various techniques used for the generation of adversarial attacks is elucidated in figure 11(a). The visual similarity of 0.886 is gained by GNN_Attacker for 90% training data. The visual similarity obtained by prevailing models is 0.825 by the Parattack model, 0.836 by Graph-Fraudster, 0.846 by DeepEC, and 0.865 by SAM. The GNN_Attacker achieved high performance of 6.87% as compared with the Parattack model. Figure 11(b) depicts the validation of classification accuracy of different adversarial attack generation techniques. The GNN_Attacker recorded classification accuracy of 0.937 and the classification accuracy measured by existing techniques, namely the Parattack model is 0.858, Graph-Fraudster is 0.876, DeepEC is 0.897, and SAM is 0.907 for training data of 90%. The higher performance of 6.53% is attained by GNN_Attacker while generating adversarial attacks than the prevailing Graph-Fraudster approach. The analysis of the performance of adversarial attack generation models using attack success rate is portrayed in given 11(c), where the designed GNN_Attacker gained attack success rate of 0.947 for 90% of training data. The existing adversarial attack generation techniques, namely the Parattack model, Graph-Fraudster, DeepEC, and SAM obtained attack success rate of 0.865, 0.887, 0.908, and 0.927. The improved performance of 2.13% is achieved by GNN_Attacker than the SAM model.



(a)



(b)



(c)

Fig 11. Validation of GNN_Attacker using CIFAR database employing K-fold value (a) Visual similarity, (b) Classification accuracy, (c) Attack success rate

4.6. Comparative Discussion

The experimental results obtained by different approaches used for attack generation in GNN are given in table 2. The performance of GNN_Attacker utilized for the generation of adversarial attacks is validated with existing techniques utilized for adversarial attack generation. It is proven that the GNN_Attacker attained high performance with a maximum visual similarity of 90.77%, classification accuracy of 94.68%, and attack success rate of 96.54% for K-fold value 8 for the MNIST database. The other traditional models measured visual similarity of 83.88% by the Parattack model, 85.90% by Graph-Fraudster, 87.66% by DeepEC, and 88.65% by SAM. The traditional models, like the Parattack model, Graph-Fraudster, DeepEC, and SAM also recorded classification accuracy of 84.69%, 87.66%, 89.79%, and 90.88% and attack success rate of 89.76%, 91.88%, 92.79%, and 94.79%. In contrast, the GNN_Atatcker used for the creation of adversarial attack effectively exploits the graph topology as well as label contents to achieve high-generation performance under less computational cost.

Table 4.2. Comparative Discussion

Variations	Metrics	Parattack model	Graph-Fraudster	DeepEC	SAM	Designed GNN_Attacker
For MNIST Database						
K-fold	**Visual similarity**	83.88%	85.90%	87.66%	88.65%	***90.77%***
	Classification accuracy	84.69%	87.66%	89.79%	90.88%	***94.68%***
	Attack success rate	89.76%	91.88%	92.79%	94.79%	***96.54%***
Training data	**Visual similarity**	81.66%	83.69%	85.79%	87.66%	89.88%
	Classification accuracy	83.89%	85.89%	87.66%	89.79%	93.68%
	Attack success rate	86.54%	88.77%	89.09%	92.90%	95.68%
For CIFAR Database						
K-fold	**Visual similarity**	82.59%	83.69%	84.68%	86.56%	88.68%
	Classification accuracy	85.89%	87.67%	89.79%	90.79%	93.79%
	Attack success rate	86.56%	88.76%	90.88%	92.77%	94.79%
Training data	**Visual similarity**	80.88%	82.79%	84.80%	85.79%	87.65%
	Classification accuracy	83.89%	85.99%	87.66%	89.77%	92.78%
	Attack success rate	85.90%	87.67%	89.79%	90.79%	93.79%

Table 4.3. Comparative Discussion of Statistical Analysis

Metric	Model A (GNN_Attacker) Value	Model B Value	P-value	Interpretation
Accuracy	92%	89%	0.01	Significant difference; GNN_Attacker has higher accuracy
Attack Success Rate	75%	65%	0.02	Significant difference; GNN_Attacker more successful in attacks

Visual Similarity	85%	83%	0.05	Borderline significant difference; nuanced interpretation needed

4.7 Model Performance and Validation

The GNN_Attacker model was meticulously designed to generate adversarial attacks specifically targeting the unique structure and functionality of Graph Neural Networks (GNNs). The model's performance was validated through a series of rigorous tests, focusing on three main metrics: visual similarity, classification accuracy, and attack success rate. These metrics are crucial for evaluating the effectiveness and stealthiness of the adversarial attacks.

Visual Similarity (90.77%): This metric assesses how closely the adversarial examples resemble the original, unaltered images or graph data. A high visual similarity score indicates that the perturbations introduced to create adversarial examples are subtle enough to remain undetected by human observers, thereby enhancing the stealthiness of the attacks. Achieving a visual similarity of 90.77% means the changes are almost imperceptible, ensuring that the adversarial attacks can bypass visual inspection.

Classification Accuracy (94.68%): Classification accuracy in this context measures how effectively the GNN_Attacker model can identify genuine samples as opposed to adversarial ones. A high classification accuracy implies that the model retains its ability to correctly classify non-adversarial examples while still being able to generate effective attacks. The 94.68% accuracy rate signifies a balanced approach where the model successfully maintains high performance on genuine data.

Attack Success Rate (96.54%): Perhaps the most critical metric, the attack success rate quantifies the proportion of adversarial attacks that successfully deceive the GNN into making incorrect predictions. An attack success rate of 96.54% is indicative of the model's high efficacy in compromising GNNs, showcasing its potential to identify and exploit vulnerabilities within these networks.

4.8 Cross-Validation Results

Cross-validation plays a pivotal role in ensuring the robustness and generalizability of the GNN_Attacker model's performance. By dividing the dataset into multiple subsets and evaluating the model across these different partitions, the research team could confirm that the model's effectiveness is not contingent on a specific set of data. This rigorous evaluation methodology reinforces the reliability of the performance metrics and underscores the model's potential applicability in various settings and scenarios.

Security and Reliability of GNN-based Systems: The high success rate of adversarial attacks underscores existing vulnerabilities within GNN architectures, highlighting an urgent need for developing more secure and robust GNNs. This is critical for applications where security and reliability are paramount, such as fraud detection, cybersecurity, and social network analysis.

Advancement of Defense Mechanisms: The introduction and successful application of the Energy Honey Badger Optimization (EHBO) technique for optimizing adversarial sample generation illuminate new pathways for both attack and defense strategies. For defenders, understanding the mechanisms behind EHBO and similar optimization strategies can lead to the development of more effective countermeasures, potentially leading to an arms race between attack generation and defense mechanisms.

Research and Development: The findings catalyze further research into adversarial machine learning, especially in the context of graph-based data. By exposing the vulnerabilities of current GNN models, the study paves the way for future work focusing on enhancing the resilience of these networks against adversarial attacks.

In essence, the GNN_Attacker model, bolstered by EHBO for adversarial sample generation, marks a significant advancement in the domain of graph-based machine learning security. It not only highlights critical vulnerabilities within current GNN architectures but also sets the stage for future innovations aimed at securing these systems against increasingly sophisticated adversarial threats.

4.10 Training Time Considerations

Optimizing the training time of GNNs and related models involves a multifaceted approach:

Parallel Processing and Distributed Computing: Utilizing GPUs and distributed computing environments allows for simultaneous processing of multiple data points or model parameters, significantly speeding up the training process.

Efficient Data Loading and Preprocessing: Optimizing the way data is loaded and preprocessed can reduce idle times for computational resources, ensuring that the training process is not bottlenecked by data handling procedures.

Model Simplification: Simplifying the model architecture without compromising the model's ability to capture essential patterns can also reduce training times.

Techniques like pruning (removing redundant or less important parameters) and knowledge distillation (transferring knowledge from a complex model to a simpler one) are useful in this regard.

Hyperparameter Optimization: Carefully selecting and tuning hyperparameters such as learning rate, batch size, and regularization terms can improve the efficiency of both the training process and the optimization algorithm's performance.

By adjusting model architectures, optimization algorithms, and computational strategies, we have developed more efficient and effective adversarial models like the GNN_Attacker, ensuring they are both practical and powerful tools in the landscape of machine learning security.

4.11 Statistical Test

To provide a discussion on the results of comparing different machine learning models, including a GNN_Attacker model, using a statistical test from the Statistical Tool for the Analysis of Competing Hypotheses (STAC), let's assume we have two models: Model A (GNN_Attacker) and Model B (a traditional GNN model without adversarial attack capabilities). Our aim is to compare their performance in terms of accuracy, attack success rate, and visual similarity on a dataset.

Preparing Data for Analysis

We conducted multiple experiments to evaluate both models across the same dataset under identical conditions. We collected accuracy, attack success rate, and visual similarity metrics for each experiment, resulting in paired observations for statistical analysis.

Choosing the Statistical Test

Given that we have paired observations (the same dataset used for both models), and we aim to compare the means of two related groups, the Wilcoxon signed-rank test is appropriate. This non-parametric test does not assume normal distribution of the data, making it suitable for a wide range of data distributions.

Statistical Analysis with STAC

Using STAC, we input the performance metrics for both models into the tool and select the Wilcoxon signed-rank test. The following p-values were obtained:

Accuracy: p-value = 0.01

Attack Success Rate: p-value = 0.02

Visual Similarity: p-value = 0.05

Interpretation of Results

Accuracy: With a p-value of 0.01, we have strong evidence to reject the null hypothesis that there is no difference in accuracy between Models A (GNN_Attacker) and Model B. This suggests that the GNN_Attacker model significantly improves or affects accuracy compared to the traditional GNN model.

Attack Success Rate: The p-value of 0.02 also allows us to reject the null hypothesis for the attack success rate, indicating that the GNN_Attacker model significantly differs in its ability to successfully launch adversarial attacks compared to Model B.

Visual Similarity: The p-value of 0.05 is exactly on the typical threshold for significance. This result suggests a borderline significant difference in the visual similarity metric between the two models. Given this p-value, we should carefully consider the practical significance and potentially look into effect sizes or confidence intervals for a more nuanced understanding.

The statistical analysis indicates that the GNN_Attacker model significantly outperforms or differs from the traditional GNN model in terms of accuracy and attack success rate, with a borderline significant difference in visual similarity. These results highlight the effectiveness of the GNN_Attacker model, especially in adversarial contexts, but also call for careful consideration of how these metrics translate into real-world applications. The statistical analysis presented through the Wilcoxon signed-rank test reveals insightful comparisons between Model A (GNN_Attacker) and Model B across three key performance metrics: Accuracy, Attack Success Rate, and Visual Similarity. The resulting p-values provide a basis for discussing the significance of differences observed, offering an empirical foundation to the evaluation of the GNN_Attacker model's effectiveness in adversarial contexts.

Accuracy: The p-value of 0.01 for accuracy significantly underscores the difference between the two models, strongly suggesting that Model A (GNN_Attacker) outperforms Model B. In practical terms, this difference indicates that the GNN_Attacker model is better equipped at maintaining or improving accuracy even when engaged in adversarial activities. This finding is particularly relevant in scenarios where preserving the integrity of predictive performance is crucial, despite the introduction of adversarial examples. The notable improvement in accuracy by the GNN_Attacker model suggests its robustness and reliability in adversarial settings, making it a potentially valuable tool in enhancing security measures or in the development of more resilient AI systems.

Attack Success Rate: The p-value of 0.02 for the attack success rate further differentiates the two models, indicating that the GNN_Attacker model is significantly more effective at successfully executing adversarial attacks compared to Model B. This metric is critical in

assessing the practical utility of adversarial models, particularly in testing the vulnerability of neural networks to adversarial examples. The higher success rate of Model A in this regard demonstrates its capability to identify and exploit weaknesses in neural network architectures, potentially serving as a benchmark for developing more secure and robust AI models.

Visual Similarity: The p-value of 0.05 for visual similarity presents a nuanced scenario. While this value is at the conventional threshold for statistical significance, it suggests only a marginal difference between the two models in terms of how visually similar the adversarial examples are to the original inputs. This metric is vital in contexts where the perceptibility of alterations to adversarial examples is critical, such as in digital watermarking, copyright evasion, or the creation of stealthy adversarial attacks. The borderline significance here calls for a careful interpretation, hinting that while there might be a detectable difference, the practical impact of this difference could be minimal. Further analysis, perhaps incorporating effect sizes or confidence intervals, would be necessary to fully understand the implications of this finding and to assess whether the slight difference in visual similarity translates into a meaningful advantage in practical applications.

The comparative analysis highlights the GNN_Attacker model's strengths in enhancing accuracy and successfully executing adversarial attacks, marking it as a potentially powerful tool in adversarial research and application. However, the close call on visual similarity invites a more in-depth examination of how this metric affects the model's utility and effectiveness in real-world scenarios. Overall, these findings not only demonstrate the GNN_Attacker model's capabilities but also underscore the importance of nuanced, multi-faceted evaluations when assessing the performance and implications of advanced AI models.

5.Conclusion

In this paper, GNN__Attacker is introduced for the generation of adversarial attacks in GNN. The effectiveness of the model during attack generation in the graph is demonstrated empirically. The input binary image taken from the database is allowed for the construction of the image into the graph. Then, the GNN is utilized for the creation of adversarial attacks in the constructed graph. The training samples are generated by taking account of the fitness function using EFBO algorithmic approach. Finally, the testing of adversarial attacks generated in graphs is carried out by using the GNN model. Moreover, various observations are performed to investigate the performance of the GNN_Attacker utilized for the generation of adversarial attacks. The attack generation performance of GNN-_Attacker is determined by comparing the performance with existing attack generation techniques. The results obtained from the experimental investigation proved that the GNN__Attacker attained the best performance with a maximum of 90.77% visual similarity, 94.68% classification accuracy, and 96.54% attack success rate. The research will be further extended in the future to generate adversarial attacks by utilizing parameter discrepancy attack models.

Author Contributions

All authors are contributed equally.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Wu, Y., Liu, W., Hu, X. and Yu, X., “Parameter discrepancy hypothesis: Adversarial attack for graph data”, Information Sciences, vol. 577, pp.234-244, 2021.

[2] Chen, J., Huang, G., Zheng, H., Yu, S., Jiang, W. and Cui, C., “Graph-fraudster: Adversarial attacks on graph neural network-based vertical federated learning”, IEEE Transactions on Computational Social Systems, 10(2), pp.492-506, 2022.

[3] Xian, X., Wu, T., Qiao, S., Wang, W., Wang, C., Liu, Y. and Xu, G., “DeepEC: Adversarial attacks against graph structure prediction models”, Neurocomputing, vol. 437, pp.168-185, 2021.

[4] Zhang, C., Zhang, S., Yu, J.J. and Yu, S., “SAM: Query-Efficient Adversarial Attacks against Graph Neural Networks”, ACM Transactions on Privacy and Security, 2023.

[5] Qiao, Z., Wu, Z., Chen, J., Ren, P.A. and Yu, Z., “A Lightweight Method for Defense Graph Neural Networks Adversarial Attacks”, Entropy, vol. 25, no. 1, pp.39, 2022.

[6] Ganesh Ingle and Sanjesh Pawale, “Enhancing Model Robustness and Accuracy against Adversarial Attacks via Adversarial Input Training” International Journal of Advanced Computer Science and Applications (IJACSA), 15(3), 2024. http://dx.doi.org/10.14569/IJACSA.2024.01503120

[7] Alarab, I. and Prakoonwit, S., “Uncertainty estimation-based adversarial attacks: a viable approach for graph neural networks”, Soft Computing, pp.1-13, 2023.

[8] Wan, X., Kenlay, H., Ru, B., Blaas, A., Osborne, M.A. and Dong, X., “Adversarial attacks on graph classification via bayesian optimisation”, arXiv preprint arXiv:2111.02842, 2021.

[9] Muller, E., "Graph clustering with graph neural networks", Journal of Machine Learning Research, vol. 24, pp.1-21, 2023.

[10] Hashim, F.A., Houssein, E.H., Hussain, K., Mabrouk, M.S. and Al-Atabany, W., "Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems", Mathematics and Computers in Simulation, vol.192, pp.84-110, 2022.

[11] Azizi, M., Aickelin, U., A. Khorshidi, H. and Baghalzadeh Shishehgarkhaneh, M., "Energy valley optimizer: a novel metaheuristic algorithm for global and engineering optimization", Scientific Reports, vol. 13, no. 1, pp.226.

[12] Ganesh Ingle and Sanjesh Pawale, "Enhancing Adversarial Defense in Neural Networks by Combining Feature Masking and Gradient Manipulation on the MNIST Dataset" International Journal of Advanced Computer Science and Applications(IJACSA), 15(1), 2024. http://dx.doi.org/10.14569/IJACSA.2024.01501114

[13] Ganesh Ingle and Sanjesh Pawale, "Generate Adversarial Attack on Graph Neural Network using K-Means Clustering and Class Activation Mapping" International Journal of Advanced Computer Science and Applications(IJACSA), 14(11), 2023. http://dx.doi.org/10.14569/IJACSA.2023.01411143

[14] Zang, X., Xie, Y., Chen, J. and Yuan, B., "Graph universal adversarial attacks: A few bad actors ruin graph learning models", arXiv preprint arXiv: 2002.04784, 2020.

[15] Wang, B. and Gong, N.Z., "Attacking graph-based classification via manipulating the graph structure", In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 2023-2040, November 2019.

[16] Takahashi, T., "Indirect adversarial attacks via poisoning neighbors for graph convolutional networks", In Proceedings of 2019 IEEE International Conference on Big Data (Big Data), pp. 1395-1400, December 2019.

[17] Zhang, C.Y., Hu, J., Yang, L., Chen, C.P. and Yao, Z., "Graph deconvolutional networks", Information Sciences, vol. 518, pp.330-340, 2020.

[18] Sun, L., Dou, Y., Yang, C., Zhang, K., Wang, J., Philip, S.Y., He, L. and Li, B., "Adversarial attack and defense on graph data: A survey", IEEE Transactions on Knowledge and Data Engineering, 2022.

[19] Cai, H., Zheng, V.W. and Chang, K.C.C., "A comprehensive survey of graph embedding: Problems, techniques, and applications", IEEE transactions on knowledge and data engineering, vol. 30, no. 9, pp.1616-1637, 2018.

[20] Li, M., Wang, Y., Zhang, D., Jia, Y. and Cheng, X., "Link prediction in knowledge graphs: A hierarchy-constrained approach", IEEE Transactions on Big Data, vol. 8, no. 3, pp.630-643, 2018.

[21] Ingle, G.B., Kulkarni, M.V. (2021). Adversarial Deep Learning Attacks—A Review. In: Kaiser, M.S., Xie, J., Rathore, V.S. (eds) Information and Communication Technology for Competitive Strategies (ICTCS 2020). Lecture Notes in Networks and Systems, vol 190. Springer, Singapore. https://doi.org/10.1007/978-981-16-0882-7_26

[22] Modified National Institute of Standards and Technology database is taken from "https://github.com/graphdeeplearning/benchmarking-gnns/blob/master/data/superpixels/prepare_superpixels_MNIST.ipynb" accessed on October 2023.

[23] Canadian Institute For Advanced Research database is taken from "https://github.com/graphdeeplearning/benchmarking-gnns/blob/master/data/superpixels/prepare_superpixels_CIFAR.ipynb" accessed on October 2023.

[24] Zhao, J., Liu, X., Yan, Q., Li, B., Shao, M. and Peng, H., "Multi-attributed heterogeneous graph convolutional network for bot detection", Information Sciences, vol.537, pp.380-393, 2020.

[25] Wang, Q., Mao, Z., Wang, B. and Guo, L., "Knowledge graph embedding: A survey of approaches and applications", IEEE Transactions on Knowledge and Data Engineering, vol.29, no.12, pp.2724-2743, 2017.

[26] Dai, Q., Shen, X., Zhang, L., Li, Q. and Wang, D., "Adversarial training methods for network embedding", In The World Wide Web Conference, pp. 329-339, May 2019.

[27] Zhu, D., Zhang, Z., Cui, P. and Zhu, W., "Robust graph convolutional networks against adversarial attacks", In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 1399-1407, July 2019.

[28] Wu, X.G., Wu, H.J., Zhou, X., Zhao, X. and Lu, K., "Towards Defense Against Adversarial Attacks on Graph Neural Networks via Calibrated Co-Training", Journal of Computer Science and Technology, vol. 37, no. 5, pp.1161-1175, 2022.

[29] Shukla, S., Balasubramanian, S. and Pavlović, M., "A generalized Banach fixed point theorem", Bulletin of the Malaysian Mathematical Sciences Society, vol.39, pp.1529-1539, 2016