# Generate Adversarial Attack on Graph Neural Network using K-Means Clustering and Class Activation Mapping

Mr. Ganesh Ingle
Department of Computer Engineering,
Vishwakarma University,
Pune, India

Dr. Sanjesh Pawale
Department of Computer Engineering,
Vishwakarma University,
Pune, India

***Abstract*—Graph Neural Networks (GNNs) have emerged as powerful tools for analyzing complex structured data, including social networks, biological networks, and recommendation systems. However, their susceptibility to adversarial attacks poses a significant challenge, especially in critical tasks such as node classification and link prediction. Adversarial attacks on GNNs can introduce harmful input graphs, leading to biased model predictions and compromising the integrity of the network. We propose a novel adversarial attack method that leverages the combination of K-Means clustering and Class Activation Mapping (CAM) to conduct subtle yet effective attacks against GNNs. The clustering algorithm identifies critical nodes within the graph, whose perturbations are likely to have a substantial impact on model performance. Additionally, CAM highlights regions of the graph that significantly influence GNN predictions, enabling more targeted and efficient attacks. We assess the efficacy of state-of-the-art GNN defenses against our proposed attack, underscoring the pressing need for robust defense mechanisms. Our study focuses on countering attacks on GNN networks by utilizing K-Means clustering and CAM to enhance the effectiveness and efficiency of the adversarial strategy. Through our observations, we emphasize the necessity for stronger security measures to safeguard GNN-based applications, particularly in sensitive environments. Furthermore, our research highlights the importance of developing robust GNNs that can withstand adversarial attacks, ensuring the reliability and trustworthiness of these models in critical applications. Strengthening the robustness of GNNs against adversarial manipulation is crucial for maintaining the security and integrity of systems that heavily rely on these advanced analytical tools. Our findings underscore the ongoing efforts required to fortify GNN-based applications, urging the research community and practitioners to collaborate in developing and implementing more robust security measures for these powerful neural network models. .**

***Keywords*—Graph neural networks; adversarial attacks; K-Means clustering; class activation mapping; robustness; defense mechanisms***

## I. Introduction

Graph Neural Networks (GNNs) have become indispensable tools in the analysis of complex graph-structured data, with applications ranging from social network analysis to recommendation systems and bioinformatics. While their ability to discern intricate relationships within graphs is advantageous, it also exposes them to potential adversarial attacks. In critical applications, such as social network analysis, adversarial manipulations could result in the propagation of misinformation or compromise user privacy [14,15,32-35]. Similarly, attacks on user-item interaction graphs in recommendation systems may lead to the tailored manipulation of content recommendations. In bioinformatics, adversarial perturbations on molecular interaction networks pose a threat to the accuracy of predictive models, especially in the identification of potential drug candidates [17-20]. The security implications outlined above necessitate a robust defense strategy to safeguard GNNs in real-world applications. Addressing the identified security issues becomes imperative to ensure the reliability of GNNs [23,25-30,36]. The proposed defense approach employs key metrics such as success rate, transferability, and computational efficiency for assessment. Preliminary evaluations reveal competitive success rates, highlighting the method's efficacy. Notwithstanding the critical need for GNN security, existing research falls short in providing comprehensive defense strategies, especially tailored to the unique challenges posed by graph data. The primary contribution of this work lies in introducing a novel adversarial attack methodology specifically designed for GNNs. This approach integrates K-Means Clustering and Class Activation Mapping to introduce structured perturbations, offering a distinctive combination of clustering and node importance information. In the realm of counterattack advancements, where existing methods face limitations in black box scenarios, the introduction of the k-Clustering Adversarial Manipulation Approach (CAMA) dataset becomes crucial. CAMA, designed to be receptive to contrasting examples, coupled with a novel approach leveraging GNNs, showcases promising potential in creating contradictory examples. This innovation contributes significantly to the development of more effective and robust counterattack techniques, marking a crucial advancement in the field of adversarial machine learning for GNNs.

## II. Background and Motivation

The escalating integration of Graph Neural Networks (GNNs) across various applications has prompted heightened concerns regarding their susceptibility to adversarial attacks. In the current landscape of research, there is a discernible dearth in the comprehensive comprehension of adversarial threats and corresponding defenses specifically tailored for GNNs. This lacuna underscores the paramount importance of fortifying GNNs against adversarial exploits to ensure their robustness in real-world deployment scenarios [2,6,7]. Our

research endeavors are motivated by the imperative to bridge the existing gaps in GNN security studies. Departing from conventional adversarial methodologies, we introduce a pioneering approach that intricately employs K-Means Clustering and Class Activation Mapping (CAM). This innovative amalgamation injects a fresh perspective into the realm of structured perturbations in GNNs, with the explicit goal of crafting more potent adversarial attacks. The principal contribution of our work lies in the conception of a novel adversarial attack methodology meticulously tailored for GNNs. The incorporation of structured perturbations, facilitated by the synergistic interplay of K-Means Clustering and Class Activation Mapping, distinguishes our approach from conventional adversarial techniques. This integration not only enriches the arsenal of adversarial methods but also provides unique insights into the vulnerabilities inherent in GNNs. Our methodology, leveraging both graph clustering and node importance information, furnishes unparalleled insights into GNN vulnerabilities. This heightened understanding empowers the creation of targeted and impactful adversarial examples, thereby propelling the advancement of adversarial research within the domain of graph-based models. Our research contributes to an elevated echelon of GNN robustness evaluation by ushering in a novel adversarial generation methodology. By seamlessly integrating K-Means Clustering and Class Activation Mapping, our approach stands as a testament to innovation in the field, offering unique perspectives and a more nuanced understanding of GNN vulnerabilities.

## III. Related Work

In recent years, white-box attack research has witnessed significant growth, focusing on three main categories: discovery-based attacks, interference attacks, and attacks causing network malfunction. These attacks are classified based on the role of the "responsible model" within a specific field, addressing challenges associated with unsupervised learning. While extensive work has been conducted on white-box attacks, this literature review specifically delves into Generative Adversarial Networks (GANs) and Convolutional Neural Networks (CNNs). These deep learning methods, known for their effectiveness, are critical in understanding and countering white-box attacks.

Among various attack methods, the Fast Gradient Sign Method (FGSM) is prominent. Operating by taking a one-step gradient of the responsible damage function, FGSM facilitates the rapid creation of "opposite" examples to the original input. Despite its popularity, FGSM doesn't always guarantee success, leading to exploration of more sophisticated strategies. Theoretical and practical challenges in unsupervised learning contribute to the complexity of white-box attack research, with a focus on GANs and CNNs offering nuanced insights and potential countermeasures.

In advancing attack strategies, an iterative approach like the Repeated Fast Gradient Method (I-FGSM) and Projected Gradient Descent (PGD) has been explored. Researchers, such as [31] and [8], iterate FGSM multiple times to create more robust adversarial examples, enhancing overall attack efficacy. Additionally, [24] introduced the Momentum-based Iterative Fast Gradient Sign Method (MIFGSM), marking a significant improvement over FGSM in terms of performance and robustness. These advancements are crucial for evaluating the robustness and security of neural networks, as researchers actively work on developing sophisticated attack and defense techniques.

In the context of white-box attacks, image-dependent targeted attacks stand out as potent threats. Recent research has seen a surge in understanding and addressing vulnerabilities in neural networks, emphasizing the urgency of exploring adversarial threats. The offensive class, as defined by [39], focuses on discovering adversarial examples within disturbance budgets to ensure misclassification by the targeted neural network. This underscores the importance of image-dependent targeted attacks and the ongoing efforts in exploring GANs and CNNs for effective countermeasures. The study in [8] introduced the Momentum Iterative Fast Gradient Method (MIFGSM), an extension that incorporates momentum into the Iterative Fast Gradient Sign Method (I-FGSM). This augmentation proves to be a substantial enhancement to the attack method, resulting in the generation of more effective and robust adversarial examples.

These competitive attack methodologies play a crucial role in assessing the reliability and security of neural networks. Ongoing research endeavors are dedicated to advancing sophisticated attack techniques, concurrently emphasizing the formulation of effective defensive strategies to mitigate the impact of such attacks.

In the domain of white-box attacks, image-dependent targeted attacks are recognized as the most potent form of adversarial threats. The exploration of adversarial attacks, particularly within the realm of white-box attacks, has garnered significant interest and witnessed notable progress.

Responsive attacks can be broadly categorized into three primary types, as delineated by [1]: Alarm-based attacks, which seek to discover counterexamples within a specified disturbance budget, inducing the neural network to misclassify with high confidence. Examples of such attacks encompass the Fast Gradient Sign Method (FGSM), Iterative Fast Gradient Method (I-FGSM), and Projected Gradient Descent (PGD). Dong et al. [8] introduced an enhanced iteration, the Momentum Iterative Fast Gradient Method (MI-FGSM).

This line of research is dedicated to minimizing interference and discovering as few counterexamples as possible to deceive neural networks without triggering alarms. Various methods have been explored for this purpose:

The research in [5] focuses on minimizing interference using simpler linear functions. Carlini and Wagner [10,22] propose a method that comes close to the goal by utilizing simpler linear functions. [3] assumes linearity near the input, contributing to the minimal interference approach.

Generative attacks employ reproductive methods to create adversarial examples. Noteworthy approaches include:

The study in [21,38] utilizes a second neural network for adversarial example generation. GANs are employed to estimate the original distribution of images, providing contrasting examples.

The study in [11] introduces the CAMA framework for addressing the creation of competing examples through extensive

adversarial manipulation of graphs. Key points include:

The study encompasses various aspects related to graph structures, node properties, and their manipulation. The approach adopted focuses on facilitating hierarchical manipulation of graph structures and node properties. In parallel, the research delves into the realm of Graph Class Activation Mapping (Graph-CAM), with the primary objective of leveraging this variant to specify node-level importance in graphic classification tasks.

A significant component of the investigation involves heuristic algorithms, demonstrating their efficacy in achieving successful performance in attributive and structural attacks. Notably, these algorithms operate under strict alarm budgets, highlighting their robustness and reliability in the context of the study.

The findings of the research underscore the significance of both node-level and subgraph-level measures. This comprehensive approach is crucial for preserving competitive interference imperceptibility. The conclusion drawn from the study emphasizes the necessity of considering measures at multiple levels to ensure a holistic understanding and effective management of graph structures and node properties in various applications.

The research focuses on systemic reciprocity attacks, specifically targeting Graph Neural Network (GNN)-based link prediction models and leveraging the SEAL algorithm. One key aspect of this investigation involves counterexample generation, with the primary task being the creation of counterexamples to optimize and deceive SEAL into generating false predictions. This strategic approach aims to understand and exploit potential vulnerabilities in the link prediction model under the influence of systemic reciprocity attacks.

An integral strength of the study lies in the utilization of SEAL's y-decomposition heuristic. This heuristic theory involves approximating graph structural properties of local subgraphs, rendering it particularly effective against systemic reciprocity attacks. By leveraging the y-decomposition heuristic, the research aims to enhance the understanding of the structural intricacies of the targeted link prediction models, thereby enabling more precise and impactful attacks. Overall, the study's emphasis on counterexample generation and the strategic use of SEAL's y-decomposition heuristic underscores its dedication to unraveling and exploiting vulnerabilities in GNN-based link prediction models under the influence of systemic reciprocity attacks.

This comprehensive approach integrates linear function-based methods, generative attacks, and the CAMA framework, emphasizing the significance of node-level and subgraph-level measures to preserve imperceptibility against adversarial manipulation. Leveraging SEAL further strengthens defense against systemic reciprocity attacks on GNN-based models.

The research in [37] underscores the relevance of competition attacks on GNN-based link prediction models. The approach gradually disrupts the network graph by manipulating its structure and utilizes a link-building mechanism along with the y-decomposition heuristic theory HERMESTIC for a more efficient competitive attack. Experimental results reveal the significant impact of planned counterattacks, posing a threat to the efficacy of SEAL league predictions, particularly with limited information about complex network diagrams.

The successful portability of attacks against various link prediction heuristics from the existing literature is highlighted, demonstrating the effectiveness and broad applicability of the proposed competitive methods. Existing competing attacks (mentioned in [3, 9, 4, 12]) often operate in a "white box" configuration, assuming full access to the machine learning model parameters. However, the passage notes the unrealistic nature of this setting when the model parameters are unknown to the attacker, emphasizing the importance of considering realistic scenarios, especially in the "black box" configuration.

Competitive attacks on link prediction algorithms showcase the need for improved resilience. Proposed approaches demonstrate effectiveness across different heuristics, emphasizing the importance of realistic scenarios, particularly in the "black box" configuration where the attacker lacks full access to model parameters. The findings contribute to a deeper understanding of challenges and potential impacts of link prediction algorithms in real-world applications.

In a point-based black box arrangement, attackers query the output of the softmax layer and obtain the final classification result (x) for a given input. Challenges arise in using traditional gradient-based approaches due to the lack of well-defined gradients in discontinuous models like decision trees. The study in [10] proposes a point-based attack that reconstructs the loss function, addressing challenges in well-defined gradients. The research in [16] introduces innovations like adaptive random gradient estimation and a well-trained autoencoder to enhance the efficiency of point-based attacks. The study in [13] proposes a point-based attack using an evolutionary algorithm, demonstrating effectiveness in both point-based and hard label black box settings.

The passage emphasizes the need for robust black box attack strategies for machine learning models, noting the limitations of existing methods. A novel approach leveraging graph neural networks (GNNs) connects networks, combining the strengths of the K-means algorithm and reinforcement learning methods. GNNs serve as an effective competitive example generation tool, bridging the gap between optimization and learning approaches in adversarial machine learning research. Advances in recent literature [10] exploring black box arrangements, specifically point-based configurations, highlight the challenges of attacking models with limited parameter information, addressing these challenges through zero-order optimization techniques [16] and evolutionary algorithms [13]. These advances contribute to the development of robust black box attack strategies, emphasizing the need for effective combinations of optimization and learning methods in adversarial machine learning research [40-44].

## IV. Methodology

The architecture of the GNN is intricately designed to closely emulate the structure of the target neural network that is the subject of compromise. This GNN operates within the context of adversarial attacks. When presented with an input image, information about its correct class, and details about an incorrect target class, the GNN engages in an iterative process. At each iteration, the GNN proposes a directional

update that aims to maximize the difference between the logits (output scores) of the incorrect target class and the correct class. The primary objective of this GNN is encapsulated in the "adversarial loss function."

The adversarial loss function serves as a crucial guiding principle for the GNN's optimization process. This function essentially formulates the goal of the GNN: to find the optimal perturbation or modification to the input image that maximizes the divergence between the logits associated with the correct class and those corresponding to the incorrect target class. The term "logits" refers to the raw, unnormalized output scores produced by the neural network before the application of a softmax function.

In essence, the GNN's architecture is tailored to navigate the input space in a way that induces misclassification. By proposing perturbations that push the decision boundaries of the target neural network, the GNN seeks to generate adversarial examples – instances where the neural network makes incorrect predictions despite minimal alterations to the input.

This adversarial approach involves an intricate interplay between the GNN's architecture, the specifics of the target neural network, and the definition of the adversarial loss function. The GNN learns to exploit the vulnerabilities and intricacies of the target model, demonstrating a nuanced understanding of the decision boundaries in the neural network it aims to compromise.

The architecture of the GNN is a sophisticated framework designed for crafting adversarial examples. It leverages the interplay of input perturbations and the intricacies of the target neural network to induce misclassification, with the adversarial loss function guiding the optimization process.

1. Initialization: - Before the evaluation process begins, the GNN is initialized with the target neural network's parameters, which are the parameters it aims to compromise. - These parameters may include weights, biases, and other network-specific parameters.

2. Forward Pass: - An input image, along with its correct class label and an incorrect target class label, is fed into the GNN. - The GNN processes this input image through its layers, which involve graph convolutional operations, and produces an output, typically in the form of class activation scores. - Class activation scores represent the GNN's predictions for each class, indicating the likelihood of the input belonging to each class.

3. Computation of Gradients (Backward Pass): - After obtaining predictions, a backward pass is performed to compute the gradients of the adversarial loss with respect to the input image. - Gradients capture the sensitivity of the adversarial loss to changes in the input image. - The adversarial loss is a measure of the difference between the logits associated with the incorrect target class and the correct class. The GNN aims to maximize this loss during the adversarial attack.

4. Adversarial Loss Maximization: - The computed gradients guide the GNN in the direction that maximally perturbs the input image to induce misclassification. - The GNN iteratively updates the input image in this direction, aiming to maximize the adversarial loss. - This process is repeated for a specified number of iterations or until a convergence criterion is met.

5. Assessment of Effectiveness: - Throughout this process, the GNN monitors the changes in the adversarial loss and observes the impact on the class predictions. - The effectiveness of the GNN is assessed based on its ability to successfully generate adversarial examples that lead to misclassification by the target neural network. - Success is measured by observing changes in the predicted class, ideally causing the neural network to classify the input image into the incorrect target class.

6. Comparison and Analysis: - The generated adversarial examples can be compared against baseline images to quantify the extent of the perturbation and evaluate the stealthiness of the attack. - Statistical measures or metrics may be employed to compare the success rates and the impact on different classes.

Below are some key points which discusses a common practice in the existing literature related to the evaluation of adversarial attacks.

The evaluation practices for adversarial attacks in image classification models are conventionally centered around assessing success rates on a predefined set of images, incorporating a specific perturbation size during the evaluation process. However, a recognized limitation in this approach raises concerns about its ability to offer a comprehensive evaluation of a model's robustness under adversarial attacks. This acknowledgment stems from the understanding that the fixed set of images, coupled with a uniform perturbation size, may not adequately capture the nuanced and diverse challenges posed by different instances within the dataset.

The highlighted variability in the robustness of images further accentuates the limitations of the current evaluation paradigm. Some images naturally exhibit resilience to attacks, leading to instances of attack failures, while others prove more susceptible, resulting in successful adversarial attacks. The inherent diversity in image responses to attacks poses a challenge to accurately discerning significant differences in success rates between various adversarial attack methods. In light of these considerations, there is a clear indication that a more comprehensive evaluation methodology is imperative. Such an approach should account for the distinct responses of different images to adversarial attacks, thereby fostering a deeper and more nuanced understanding of the model's vulnerability across diverse scenarios.

### *A. Proposed Architecture*

In the context of the MNIST dataset, each image is conceptualized as a node in a graph. The relationships between these nodes are established through edges, which have the potential to capture spatial or contextual connections between pixels.

#### *1. Graph Neural Network (GNN) Architecture*

The architecture of a Graph Neural Network (GNN) typically comprises multiple graph convolutional layers. These layers play a crucial role in processing information from neighboring nodes and subsequently updating node representations based on the received information.

*2. Forward Propagation Process*

The initialization phase involves assigning each node, representing an image in MNIST, with a feature vector typically derived from pixel values. The forward propagation within the Graph Neural Network (GNN) includes the repeated application of graph convolutional layers. The mathematical representation of this process is given by the equation:

$$h_v^{(l+1)} = \sigma\left(\sum_{u \in N(v)} W^{(l)} h_u^{(l)} + b^{(l)}\right) \quad (1)$$

Here, $h(l)_v$ denotes the representation of node $v$ at layer $l$, $N(v)$ represents the neighbors of node $v$, $W(l)$ is the weight matrix at layer $l$, $b(l)$ is the bias term, and $\sigma$ is the activation function.

The aggregation step involves combining information from neighboring nodes based on the graph structure, ensuring that each node's representation incorporates information from its immediate context. The final layer of the GNN produces the output, utilized for various tasks such as classification or regression. The underlying mathematical insight lies in the graph convolutional operation, where each node's representation is updated by considering the representations of its neighbors, weighted by learned parameters. These parameters $W(l)$ and $b(l)$ are learned during the training process through backpropagation and optimization algorithms.

*3. Training*

The training process of Graph Neural Networks (GNNs) involves feeding labeled data through the network, computing a loss function, and updating parameters using optimization techniques like gradient descent. This iterative training approach allows the network to learn and adjust its parameters to improve its performance on the given task.

The utilization of this architecture enables GNNs to effectively capture and leverage the inherent graph structure in the MNIST dataset. This capability makes GNNs a potent tool for tasks such as image classification, where understanding and utilizing relationships between images are crucial for accurate predictions.

It's important to note that the specifics of the GNN architecture and mathematical operations may vary based on the exact implementation and variations in GNN models. Different approaches and variations in the model design can influence how the network processes information and learns from the input data.

*4. Proposed Architecture Diagram*

The provided Fig. 1 describes the proposed architecture diagram of the approach, illustrating the GNN layer update, class activation scores, and K-Means clustering steps.

*B. Graph Representation:*

Each image in the MNIST dataset is symbolically depicted as a node within a graph, resulting in a total number of nodes equivalent to the dataset's image count. The structural foundation of this graph is established by edges, which serve as connectors between nodes. The interpretation of these connections is flexible, allowing the model or designer to attribute meaning to the relationships encoded within the graph. The edges, in particular, may encapsulate spatial connections reflective of the pixel arrangements in the images. One conceivable interpretation involves connecting pixels in close proximity within an image. By doing so, the graph becomes a representation capable of capturing the spatial relationships embedded in the pixel structure of the images, contributing to a richer and more nuanced understanding of the dataset.

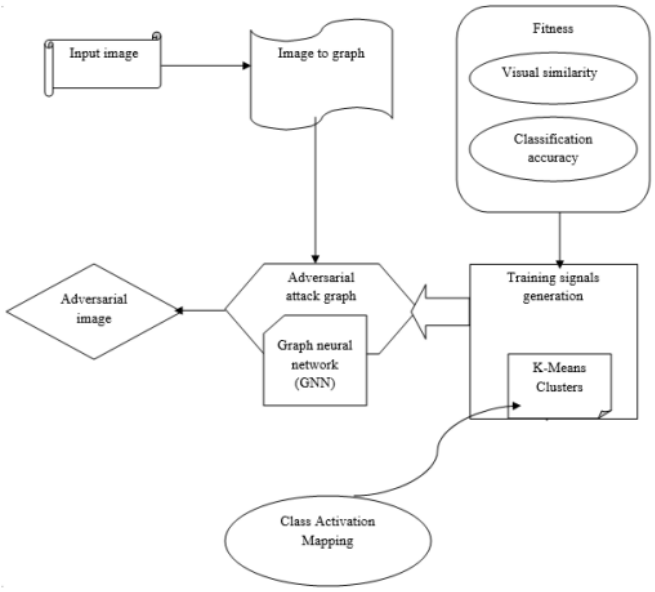


Fig. 1. Architecture of proposed approach.

The utilization of a Graph Neural Network (GNN) implies a deliberate leveraging of the underlying graph structure for the processing and analysis of data. GNNs, typically employed for such tasks, function by aggregating and updating node representations through graph convolutional layers. In the specific context of MNIST, this entails considering the pixel values of neighboring nodes when updating the representation of a given node, emphasizing the importance of spatial relationships in image data.

In the realm of contextual connections, the edges in the graph extend beyond spatial relationships, potentially representing contextual connections between images based on shared similarities or patterns within the dataset. GNNs, equipped with the capacity to learn and capture intricate contextual information from these graph connections, prove particularly advantageous for tasks like image classification, where understanding the context of an image significantly influences performance.

The structured graph representation aligns with the inherent structure of the images in the MNIST dataset, with nodes symbolizing individual images and edges encapsulating relationships between them. Depending on the GNN's design, the graph may exhibit dynamic characteristics, with edges adapting during the training process based on learned relationships.

This approach facilitates the flexibility to capture both spatial and non-spatial features, depending on how edges are defined. The graph's adaptive learning, inherent to GNNs, allows the model to dynamically adjust to the unique characteristics of the MNIST dataset, effectively leveraging both spatial and contextual information for enhanced performance in image-related tasks. In essence, the graph representation within the context of a GNN provides a structured and adaptable framework for comprehensively understanding the relationships between images, crucial for tasks like image classification.

### C. Graph Neural Network (GNN) Architecture:

The primary purpose of graph convolutional layers in a GNN is to process information from neighboring nodes and update node representations. A typical GNN architecture involves multiple graph convolutional layers stacked on top of each other. Each layer processes information from the neighboring nodes of each node in the graph. Neighboring nodes are determined based on the edges in the graph.The forward propagation begins with the initialization of node representations. Each node is initialized with a feature vector, often derived from the input data (e.g., pixel values for images).The forward propagation involves the repeated application of graph convolutional layers. In each layer, information from neighboring nodes is aggregated to update the node representations.

Mathematically, the update process for the representation of a node $v$ at layer $l+1$ can be represented as:

$$h_v^{(l+1)} = \sigma\left(\sum_{u \in N(v)} W^{(l)} h_u^{(l)} + b^{(l)}\right) \tag{2}$$

Here, $h_v^{(l)}$ is the representation of node $v$ at layer $l$, $N(v)$ represents the neighbors of node $v$, $W^{(l)}$ is the weight matrix at layer $l$, $b^{(l)}$ is the bias term, and $\sigma$ is the activation function.

The aggregation step in a Graph Neural Network (GNN) is a crucial process that involves consolidating information from neighboring nodes based on the underlying graph structure. This ensures that each node's representation is enriched with insights from its immediate context. Following the aggregation, the updated node representations capture both the inherent features of the node itself and information gleaned from its neighbors. This dual consideration enables the GNN to factor in both local and global information during the learning process.

Moving to the output layer, it serves as the final stage where the GNN produces task-specific outputs, such as classifications or regression predictions. The operations within this layer are tailored to the specific task at hand, involving transformations of the learned representations into meaningful predictions.

The learning process involves the adaptation of parameters, including weight matrices ($W^{(l)}$) and biases ($b^{(l)}$), through techniques like backpropagation and optimization algorithms. This adaptive learning mechanism empowers the GNN to dynamically adjust its parameters based on the patterns and relationships discerned from the graph-structured data.

The flexibility and adaptability of the GNN architecture further contribute to its efficacy. The graph structure, inherently flexible, allows the model to adapt to various graph structures, with edges and connections potentially changing during training based on the learned relationships. The intermediate layers of graph convolution provide task-agnostic representations of nodes, rendering the GNN suitable for a spectrum of graph-related tasks.

A GNN equipped with multiple graph convolutional layers systematically processes information from neighboring nodes, updating node representations, and adeptly capturing both local and global context. The adaptive learning mechanism ensures the GNN's proficiency in learning and adjusting parameters, enabling effective modeling of relationships within graph-structured data.

### D. Forward Propagation Process:

In the context of the MNIST dataset, the graph representation treats each image as a node. The initialization process begins by assigning a feature vector to each node, where this vector serves as the initial representation of the corresponding image.

The source of features for each node lies in the pixel values of the corresponding image. These pixel values, which convey intensity or color information at different locations within the image, are typically organized in a grid-like structure. Depending on the design of the Graph Neural Network (GNN) and the specific task requirements, preprocessing steps may be applied to the pixel values. These steps could include normalization to a specific range or other transformations.

The feature vector itself is a numerical representation that encapsulates essential characteristics of the image. Each element of the vector corresponds to a specific feature or pixel value, and the dimensionality of this vector is determined by the number of elements it contains. This dimensionality is a critical factor influencing the GNN's capacity to capture and process information effectively.

During the training process, the parameters of the GNN, including those related to the initialization of the feature vectors, are learnable and adjusted based on observed patterns in the data. The adaptability of the GNN allows it to dynamically learn and update node representations as it processes information and identifies relevant patterns within the graph-structured data.

In task-specific scenarios, the initialization process may be tailored to the particular task the GNN is designed for. For instance, in image classification tasks, the feature vector should be crafted to capture characteristics pertinent to distinguishing between different classes of images. The quality of these initial feature vectors significantly influences the GNN's learning process, as well-initialized representations provide a robust foundation for the model to build upon during training.

The update process for the representation of a node $v$ in a graph neural network (GNN) at layer $l+1$. Let's break down the components of the equation:

$$h_v^{(l+1)} = \sigma\left(\sum_{u\in\mathcal{N}(v)} W^{(l)} h_u^{(l)} + b^{(l)}\right) \tag{3}$$

$h_v^{(l+1)}$: This is the representation of node $v$ at layer $l+1$. It captures the information about the node after the application of the graph convolutional layer.

$\mathcal{N}(v)$: Represents the neighbors of node $v$ in the graph. The sum is taken over all neighboring nodes.

$W^{(l)}$: The weight matrix at layer $l$. This matrix contains learnable parameters that are adjusted during the training process to capture the relationships between nodes.

$h_u^{(l)}$: The representation of a neighboring node $u$ at layer $l$. It contributes to the update of the central node's representation.

$b^{(l)}$: The bias term at layer $l$. It provides an additional learnable parameter that influences the node representation.

$\sigma$: The activation function. It introduces non-linearity to the model. Common choices include the sigmoid ($\sigma(x) = \frac{1}{1+e^{-x}}$), hyperbolic tangent ($\sigma(x) = \tanh(x)$), or Rectified Linear Unit (ReLU) ($\sigma(x) = \max(0, x)$).

The equation describes the aggregation of information from neighboring nodes using the weight matrix $W^{(l)}$, and the result is passed through an activation function $\sigma$. This operation is a fundamental step in the forward propagation of a GNN, allowing the model to capture and update node representations based on the graph structure.

This mathematical formulation is crucial for understanding how information flows through the graph convolutional layers, enabling the GNN to learn hierarchical and contextual representations of nodes in the graph.

The aggregation involves combining information from neighboring nodes based on the graph structure. This step ensures that each node's representation incorporates information from its immediate context.

$$\text{Aggregated Information} = \sum_{u\in\mathcal{N}(v)} \text{Weight}\times\text{Representation of } u \tag{4}$$

Here, $\mathcal{N}(v)$ represents the set of neighbors for the central node $v$.

*1) Output Layer:* The graph convolutional operation at layer $l$ can be mathematically represented as:

$$h_v^{(l+1)} = \sigma\left(\sum_{u\in\mathcal{N}(v)} W^{(l)} h_u^{(l)} + b^{(l)}\right) \tag{5}$$

$h_v^{(l+1)}$ : Representation of node $v$ at layer $l+1$

$\mathcal{N}(v)$ : Set of neighbors of node $v$ in the graph

$W^{(l)}$ : Weight matrix at layer $l$

$h_u^{(l)}$ : Representation of neighboring node $u$ at layer $l$

$b^{(l)}$ : Bias term at layer $l$

$\sigma$ : Activation function

The parameters $W^{(l)}$ and $b^{(l)}$ are learned during the training process through backpropagation and optimization algorithms.

Training involves feeding labeled data through the network, computing a loss function, and updating the parameters using optimization techniques like gradient descent.

This overall architecture enables GNNs to capture and leverage the inherent graph structure in the MNIST dataset, providing a powerful tool for tasks such as image classification. The specifics of the architecture and mathematical operations may vary based on the exact implementation and variations in GNN models.

For a GNN layer $l$ with node representations $H^{(l)}$ and edge connectivity $E$, the update equation is given by:

$$H^{(l+1)} = \text{Aggregator}(H^{(l)}, E) \tag{6}$$

where, Aggregator is a function that aggregates information from neighboring nodes and edges.Class activation scores ($S_c$) are computed for each node in the graph. Let $H^{(1)}$ represent the node representations after the first GNN layer. For each class $c$, the class activation score $S_c[i]$ for each node $i$ is computed using the ReLU activation function and the corresponding class weight $W_{1c}$:

$$S_c[i] = \text{ReLU}(H^{(1)}[i] \cdot W_{1c}) \tag{7}$$

K-means clustering is applied to group nodes based on their activation scores. The mathematical modeling for K-means clustering involves finding $K$ cluster centers $\mu_k$ that minimize the sum of squared distances to their assigned data points:

$$\arg\min_{\mu_1,\mu_2,\ldots,\mu_K} \sum_{i=1}^{N}\sum_{k=1}^{K} \delta(i,k) \cdot \|S_c[i] - \mu_k\|^2 \tag{8}$$

where, $N$ is the total number of nodes, $\delta(i,k)$ is the Kronecker delta, $S_c[i]$ is the class activation score for node $i$ and class $c$, and $\mu_k$ is the cluster center for cluster $k$. The objective of K-means clustering is to find $K$ cluster centers $\mu_k$ that minimize the sum of squared distances between each node's class activation score $S_c[i]$ and the cluster centers $\mu_k$:

$$\arg\min_{\mu_1,\mu_2,\ldots,\mu_K} \sum_{i=1}^{N}\sum_{k=1}^{K} \delta(i,k) \cdot \|S_c[i] - \mu_k\|^2 \tag{9}$$

Here, $K$ is the number of clusters, $\delta(i,k)$ is the indicator function, $S_c[i]$ is the class activation score for node $i$ and class $c$, and $\mu_k$ is the cluster center for cluster $k$.

K-means clustering proceeds through the following iterative steps:

*Initialization:* Randomly initialize $K$ cluster centers $\mu_k$.

*2) Assignment:* Assign each node to the nearest cluster center based on the Euclidean distance between its class activation score $S_c[i]$ and the cluster centers $\mu_k$:

$$\delta(i,k) = \begin{cases} 1 & \text{if } k = \arg\min_j \|S_c[i] - \mu_j\| \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

*3) Update Centers:* Recalculate the cluster centers $\mu_k$ as the mean of the class activation scores of nodes assigned to each cluster:

$$\mu_k = \frac{\sum_{i=1}^{N} \delta(i,k) \cdot S_c[i]}{\sum_{i=1}^{N} \delta(i,k)} \tag{11}$$

*4) Convergence:* Repeat the assignment and update steps until convergence, where minimal change in cluster assignments or cluster centers indicates stability.

- Cluster Centers $\mu_k$: Representative points for each cluster, adjusted to minimize the sum of squared distances. - Node Assignment $\delta(i,k)$: Indicator function assigning nodes to the cluster with the closest center. - Objective Function: Minimizes the sum of squared distances, encouraging similar class activation scores within clusters. - Initialization and Convergence: Sensitive to initialization; iterative nature ensures convergence to a stable solution.

*Step 2.1: Initialize Cluster Centers*

First, $K$ initial cluster centers $\mu_k$ are randomly chosen from the data points. These centers represent the initial guess of the cluster centroids.

*Step 2.2: Assign Nodes to Clusters*

Each node is assigned to the cluster whose cluster center is closest to it. This assignment is based on the Euclidean distance between the class activation scores of the node ($Sc[i]$) and the cluster center ($\mu_k$):

$$\delta(i,k) = \begin{cases} 1 & \text{if node } i \text{ is assigned to cluster } k, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

$$k_{\text{assigned}} = \arg\min_k \|Sc[i] - \mu_k\|^2 \tag{13}$$

*Step 2.3: Update Cluster Centers*

After assigning all nodes to clusters, the cluster centers are updated by taking the mean of the class activation scores of the nodes within each cluster:

$$\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} Sc[i] \tag{14}$$

represents the set of nodes in cluster $k$.

*Step 2.4: Repeat Assignment and Update*

Steps 2.2 and 2.3 are repeated until convergence. At each iteration, nodes are reassigned to clusters based on updated cluster centers, and centers are recalculated based on new assignments.

*Identifying Target Cluster*

For the target class $\text{target}_c$, the cluster $\text{target}_C$ with the highest sum of class activation scores for that target class is identified:

$$\text{target} = \arg\max_k \sum_{i \in \text{Cluster}_k} S_c^{\text{target}}[i] \tag{15}$$

Here, $\text{target}_C$ is the cluster containing nodes with high activation scores for the target class $\text{target}_c$.

- Cluster Initialization ($\mu_k$): Represents the initial guess of cluster centroids. - Node Assignment ($\delta(i,k)$): Assigns each node to the cluster with the closest center. - Update Centers ($\mu_k$): Recalculates cluster centers based on the mean of class activation scores. - Iteration and Convergence: Repeats assignment and update steps until convergence. - Identifying Target Cluster: Locates the cluster with the highest sum of class activation scores for the target class.

To create an adversarial attack, we perturb the characteristic vectors of selected target nodes. The purpose is to make small changes to the feature vectors so that the GNN misclassifies selected target nodes. By adding a small amount of noise to the feature vectors, we aim to push the decision limits of the GNN, causing incorrect predictions.

For each selected target node $i$, we introduce a noise vector $\epsilon_i$ to perturb its feature vector

$$X_{\text{perturbed}}[i] = X_{\text{original}}[i] + \epsilon_i \tag{16}$$

Here: - $X_{\text{original}}[i]$ is the original feature vector of node $i$. - $X_{\text{perturbed}}[i]$ is the perturbed feature vector of node $i$. - $\epsilon_i$ is the noise vector for node $i$.

The noise vector $\epsilon_i$ is typically drawn from a small distribution around zero, such as a normal distribution with mean 0 and a small standard deviation $\sigma$. The standard deviation $\sigma$ controls the magnitude of the perturbation. Modest changes in the feature space during forward propagation might result in various activations and node representations in the GNN, changing the decision bounds and leading to misclassification of the target nodes.

To analyze the impact of perturbation on the GNN's decision, we can perform a Taylor expansion analysis. Let $F(X_i)$ represent the GNN's output (e.g., class probabilities) for node $i$ with the feature vector $X_i$. The Taylor expansion can be expressed as:

$$F(X_{\text{perturbed}}[i]) = F(X_{\text{original}}[i]) + \nabla F(X_{\text{original}}[i]) \cdot \epsilon_i + O(\|\epsilon_i\|^2) \tag{17}$$

Here: - $\nabla F(X_{\text{original}}[i])$ represents the gradient of $F$ with respect to $X_{\text{original}}[i]$. - $O(\|\epsilon_i\|^2)$ represents the higher-order terms that involve the square of the perturbation $\epsilon_i$.

From the Taylor expansion, we can observe that the perturbation $\epsilon_i$ contributes to the change in the GNN's output, and the gradient $\nabla F(X_{\text{original}}[i])$ indicates the sensitivity of the model's output to perturbations. The specific expression for $\nabla F(X_{\text{original}}[i])$ depends on the GNN architecture and the specific layers used.

## V. Algorithm

**Input:**

- GNN model (previously trained on the MNIST dataset)
- MNIST dataset (with labeled images)
- Target class $c_{\text{target}}$ (the class you want to misclassify)

- Number of clusters $K$ for K-means
- Number of target nodes to select from each cluster $N$
- Standard deviation of noise $\sigma$ for perturbation
- Scaling factor $\alpha$ for perturbation control

**Output:**

- Perturbed graph with target nodes modified for the adversarial attack

1) Preprocess the MNIST Dataset: Load the MNIST dataset and preprocess the images (e.g., normalize pixel values to the range [0, 1]).
2) Compute Class Activation Scores: Perform forward propagation on the graph to obtain the output of the first GNN layer $H^{(1)}$. For each class $c$ in the MNIST dataset, compute the class activation scores $S_c$ using the formula:

$$S_c = \text{ReLU}(H^{(1)} \cdot W_{1c})$$

where $W_{1c}$ is the weight matrix corresponding to class $c$, and ReLU is the Rectified Linear Unit activation function.
3) Apply K-means Clustering: Apply K-means clustering to the class activation scores $S_c$ to group nodes into $K$ clusters based on their activation patterns. Obtain the cluster assignments for each node in the graph.
4) Select Target Nodes: Identify the cluster $C_{\text{target}}$ with the highest sum of class activation scores for the target class $c_{\text{target}}$:

$$C_{\text{target}} = \arg\max_k \sum_{i \in \text{Cluster}_k} S_{c_{\text{target}}}[i]$$

From the cluster $C_{\text{target}}$, select the $N$ target nodes with the highest class activation scores for class $c_{\text{target}}$. If $N$ is larger than the number of nodes in $C_{\text{target}}$, select all nodes in $C_{\text{target}}$.
5) Perturb Node Features: For each selected target node $i$, compute the noise vector $\epsilon_i$ from a normal distribution with mean 0 and standard deviation $\sigma$:

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$$

where $I$ is the identity matrix. Rescale the noise vector $\epsilon_i$ with the scaling factor $\alpha$ to control the strength of the perturbation:

$$\epsilon_i' = \alpha \cdot \epsilon_i$$

Perturb the feature vector of each selected target node:

$$\text{Perturbed feature vector}[i] = \text{clip}(\text{Original feature vector}[i] + \epsilon_i, 0, 1)$$

where clip ensures that the perturbed features stay within the valid range [0, 1].
6) **Reevaluate GNN:** Reevaluate the GNN model on the modified graph with the perturbed features. Check if the target nodes are now misclassified as class $c_{\text{target}}$.
7) Evaluate Attack Success Rate: Measure the success rate of the attack by calculating the percentage of misclassified target nodes out of the total number of selected target nodes.

## VI. Experimental Set Up

*1. Dataset Description:*

The MNIST dataset is a widely recognized and extensively used collection of grayscale images, each depicting handwritten digits ranging from 0 to 9. Each image is formatted as a 28x28 pixel grid, resulting in a total of 784 pixels per image. The pixel values range from 0 to 255, with 0 representing black and 255 representing white. This dataset is partitioned into two primary subsets: a training set with 60,000 images and a test set with 10,000 images. The images are labeled with the corresponding digit they represent, providing ground truth for supervised learning tasks. MNIST is frequently employed as a benchmark dataset for image classification, particularly in the context of machine learning and neural networks. Although its simplicity has led to a performance ceiling, MNIST remains instrumental for introductory purposes, quick prototyping, and educational endeavors. Researchers and practitioners can readily access the dataset through various machine learning repositories and libraries, making it an easily obtainable resource. Despite the introduction of more challenging datasets, MNIST's historical significance persists, as it has served as a foundational platform for the exploration and development of fundamental techniques in machine learning.

*2. GNN Architecture:*

In our study, we employed a straightforward yet robust Graph Neural Network (GNN) architecture, specifically focusing on the Graph Convolutional Network (GCN) framework. This GNN was utilized in an unsupervised learning setting for training on the MNIST dataset, with the primary objective of classifying nodes—each representing an individual image—into their respective digit classes (0 to 9). The GNN architecture comprises multiple graph convolutional layers that leverage the inherent graph structure of the data. Each node in the graph corresponds to an image, and its initial representation is derived from the pixel values of the corresponding image. Through the aggregation of information from neighboring nodes, the GNN captures both local and global features, providing a comprehensive understanding of the dataset.

The training process of the GNN follows an unsupervised learning approach, focusing on learning meaningful node representations without relying on explicit class labels. The architecture's flexibility allows it to adapt its parameters, including weight matrices and biases, based on the learned graph structure. Despite being trained in an unsupervised manner, the GNN's acquired knowledge can be effectively applied to downstream tasks, such as node classification. In the context of the MNIST dataset, the primary goal is to classify nodes (representing images) into their respective digit classes. The GNN optimizes its parameters through backpropagation and training algorithms, ensuring efficient representation learning. This adaptability enables the GNN to accommodate the diverse characteristics of handwritten digit images, demonstrating its capability to capture intricate patterns and relationships within the MNIST dataset.

*3. Forward Propagation:*

Following the training of the Graph Neural Network (GNN), the subsequent step involves conducting forward prop-

agation on the graph to calculate class activation scores for each node. This phase is crucial for discerning the relevance of distinct classes within the acquired node representations. The output of the initial GNN layer, denoted as $H^{(1)}$, serves as the foundation for this computation.

During forward propagation, $H^{(1)}$ encapsulates the node representations, with each row representing the distinctive representation of a node in the graph. Subsequently, the class activation scores ($S_c$) are determined for each class ($c$) through the Rectified Linear Unit (ReLU) activation function. The mathematical expression governing this process is articulated as $S_c = \text{ReLU}(H^{(1)} \cdot W_{1c})$, where $W_{1c}$ denotes the weight matrix specific to class $c$.

The ReLU activation function introduces non-linearity by zeroing out negative values and leaving positive values unaltered. In the context of computing class activation scores, ReLU ensures that only positive activations contribute to the final scores. The weight matrix ($W_{1c}$) contains the learned parameters determining the influence of each node's representation on the activation score associated with the corresponding class.

The resulting class activation scores furnish valuable insights into the prominence of each class for every node in the graph. Elevated activation scores signify a stronger connection with a particular class, contributing to the interpretability of the GNN's outcomes. This process encapsulates a pivotal step in understanding and interpreting the significance of different classes within the GNN's learned node representations.

*4. K-means Clustering:*

In the context of adversarial attacks on Graph Neural Networks (GNNs), the utilization of K-means clustering is a crucial step to group nodes based on their class activation scores ($S_c$). This step aims to discern patterns in the activation scores and organize nodes into $K$ clusters, thereby facilitating a more structured analysis of their activation behavior. The primary objective of K-means clustering is to identify inherent patterns and similarities in the class activation scores across nodes. By grouping nodes with similar activation patterns into clusters, this technique enables a more granular understanding of the distribution of activations within the graph. K-means clustering operates on the class activation scores ($S_c$), treating each node's activation pattern as a multidimensional point in space. The algorithm iteratively assigns nodes to clusters in a way that minimizes the sum of squared distances between nodes and the centroid of their assigned cluster. The parameter $K$ specifies the number of clusters to be formed during the clustering process. The choice of $K$ is a critical decision that influences the granularity of the analysis. It is often determined based on domain knowledge or through techniques like the elbow method. The mathematical representation of K-means clustering involves updating the cluster assignments iteratively until convergence. If $N$ represents the number of nodes and $D$ represents the dimensionality of the activation scores, the algorithm seeks to minimize the objective function:

$$\arg\min_C \sum_{k=1}^{K} \sum_{i \in C_k} ||S_c[i] - \mu_k||^2 \tag{18}$$

where, $C$ denotes the cluster assignments, $C_k$ represents the nodes in cluster $k$, $S_c[i]$ is the activation score for node $i$, and $\mu_k$ is the centroid of cluster $k$. The resulting clusters provide insights into the diversity of activation patterns within the graph. Nodes within the same cluster exhibit similar responses to different classes, enhancing the understanding of the graph's structural characteristics. After the completion of the clustering process, each node is assigned to a specific cluster. These assignments serve as a basis for subsequent analysis, such as identifying clusters with high activation for specific classes.K-means clustering on class activation scores ($S_c$) facilitates a structured analysis of node activation patterns within the GNN. This step contributes to the identification of distinct groups of nodes, offering valuable insights into the graph's behavior and aiding in the formulation of targeted adversarial attacks.

*5. Target Node Selection:*

The process of selecting target nodes is a crucial step in the adversarial attack methodology, specifically after identifying the cluster ($C_{\text{target}}$) with the highest activation sum for the target class ($c_{\text{target}}$). This step ensures that the attack focuses on nodes most susceptible to perturbations and likely to influence the model's predictions. Identification of Target Cluster ($C_{\text{target}}$):The K-means clustering algorithm is employed to group nodes based on their activation patterns, resulting in various clusters. $C_{\text{target}}$ is the cluster with the highest sum of class activation scores for the target class ($c_{\text{target}}$).

Selection of Target Nodes ($N$ nodes): From $C_{\text{target}}$, $N$ nodes are chosen to be the target nodes for the adversarial attack. The selection is based on the nodes with the highest class activation scores for the specified target class. If $N$ exceeds the number of nodes in $C_{\text{target}}$, all nodes in $C_{\text{target}}$ are included as target nodes.

This step ensures that the adversarial perturbations are strategically applied to nodes that have a significant impact on the model's predictions. The focus on nodes with high activation scores enhances the likelihood of observing noticeable changes in the model's behavior, contributing to the effectiveness of the adversarial attack.

*6. Noise Vector Computation:*

The computation of the noise vectors ($\epsilon_i$) is a pivotal step in the adversarial attack process, aiming to introduce controlled perturbations to the feature vectors of selected target nodes. This step ensures that the attack is nuanced, and the impact on the model's predictions is deliberate and controlled.For each selected target node ($i$), a noise vector ($\epsilon_i$) is generated from a normal distribution with a mean of 0 and a standard deviation ($\sigma$). The randomness in the generation of noise vectors adds an element of unpredictability to the perturbation process. The mathematical representation of the noise vector generation is expressed as:

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$$

Here, $\mathcal{N}$ represents the normal distribution, 0 is the mean, $\sigma$ is the standard deviation, and $I$ is the identity matrix.The standard deviation ($\sigma$) plays a crucial role in controlling the strength of the perturbation. A higher $\sigma$ value results in more

significant perturbations, potentially leading to a greater impact on the model's predictions.

To further control the strength of the perturbation, the generated noise vector ($\epsilon_i$) is rescaled by a scaling factor ($\alpha$). The rescaled noise vector ($\epsilon'_i$) is given by:

$$\epsilon'_i = \alpha \cdot \epsilon_i$$

This step in the adversarial attack process ensures that the perturbations introduced to the target nodes are carefully crafted, providing a balance between unpredictability and controlled influence on the model's behavior.

*7. Noise Vector Rescaling:*

The noise vector ($\epsilon_i$) generated for each selected target node undergoes a crucial step of rescaling to control the strength of the perturbation. This rescaling operation, governed by a scaling factor ($\alpha$), plays a pivotal role in determining the impact of the perturbation on the feature vectors of the target nodes. The rescaling of the noise vector is expressed mathematically as:

$$\epsilon'_i = \alpha \cdot \epsilon_i$$

Here, $\epsilon_i$ represents the generated noise vector, $\alpha$ is the scaling factor, and $\epsilon'_i$ is the rescaled noise vector. The scaling factor ($\alpha$) acts as a control parameter for the strength of the perturbation. A higher value of $\alpha$ amplifies the impact of the perturbation, influencing the modified feature vectors of the target nodes to a greater extent. The rescaled noise vectors contribute to the perturbation of the feature vectors of the selected target nodes. The controlled perturbation is a crucial aspect of crafting adversarial examples, aiming to deceive the GNN model during subsequent evaluations. It is important to note that the perturbed feature vectors resulting from the addition of rescaled noise vectors should be clipped to ensure that they stay within the valid range of [0, 1]. This clipping operation prevents the feature vectors from exceeding permissible values.

Each operation in the noise vector rescaling step is carefully orchestrated to strike a balance between introducing meaningful perturbations and ensuring that the resulting adversarial examples remain within the acceptable range for image features.

*8. Feature Vector Modification:*

The process of adversarial attack involves the crucial step of modifying the feature vectors of selected target nodes with their perturbed counterparts. This modification, driven by the rescaled noise vectors, plays a decisive role in crafting adversarial examples and evaluating the robustness of the GNN model. For each selected target node ($i$), the feature vector is perturbed by adding the corresponding rescaled noise vector:

$$\text{Perturbed feature vector}[i] = \text{clip}(\text{Original feature vector}[i] + \epsilon'_i, [0, 1]) \tag{19}$$

Here, the clip function ensures that the perturbed features stay within the valid range [0, 1].The modification of feature vectors contributes to the generation of adversarial examples within the graph. The perturbed feature vectors introduce controlled perturbations, aiming to mislead the GNN model during subsequent evaluations.The modified graph, incorporating perturbed feature vectors, is then used to reevaluate the GNN model. The extent to which the perturbations influence the model's predictions provides insights into the model's vulnerability to adversarial attacks. The ultimate goal is to assess whether the perturbed target nodes are now misclassified as the specified target class ($c_{\text{target}}$). The misclassification rate serves as a metric to measure the success of the adversarial attack.

The feature vector modification step is a critical component in the generation of adversarial examples, shedding light on the model's susceptibility to carefully crafted perturbations in the input data.

*9. Re-evaluation of GNN Model:*

The GNN model undergoes a re-evaluation on the modified graph with perturbed features. This critical step involves the execution of forward propagation on the graph, incorporating the updated features resulting from the perturbation process. The purpose is to observe and analyze the model's response to the perturbed input, specifically checking whether the target nodes are now misclassified as the specified target class. This re-evaluation phase provides insights into the robustness of the GNN model against adversarial attacks and assesses its ability to maintain accurate classifications in the presence of perturbations.

*10. Misclassification Check:*

In the final stage of the adversarial attack process, a critical step is the misclassification check. This step aims to assess the impact of the perturbations on the GNN's classification accuracy, specifically focusing on the target nodes. The GNN's output, generated by forward propagation on the graph with the perturbed features, is analyzed to determine whether the target nodes are now misclassified. Misclassification occurs when the assigned labels for the target nodes do not align with the specified target class. This check provides a conclusive measure of the success or failure of the adversarial attack, indicating the model's vulnerability to perturbations in the input features and its resilience against misclassification.

*11. Success Rate Measurement:*

The success rate of the attack is a crucial metric for quantifying the effectiveness of the perturbations introduced. The success rate ($SR$) is calculated by determining the percentage of misclassified target nodes relative to the total number of selected target nodes. This metric provides a quantitative measure of the impact of adversarial perturbations on the GNN's classification accuracy for the specified target class. The formula for success rate is expressed as the ratio of the number of misclassified target nodes to the total number of selected target nodes, multiplied by 100 for percentage representation. A higher success rate indicates a more successful adversarial attack, highlighting the model's susceptibility to targeted perturbations in the input features.

*12. Fine-tuning:*

In the fine-tuning phase of the adversarial attack, the goal is to systematically optimize the attack strategy by iterating through Steps 5 to 9 with varied values of $K$, $N$, $\sigma$, and

$\alpha$. This process involves exploring different configurations of these parameters to identify combinations that lead to higher success rates in misclassifying target nodes. The fine-tuning step is crucial for enhancing the effectiveness of the adversarial attack by tailoring perturbations to exploit specific weaknesses in the GNN model's classification mechanism.

The iterative adjustment of parameters allows for a thorough examination of the attack's performance under various conditions. By monitoring and comparing success rates across iterations, the fine-tuning phase aims to identify parameter values that consistently result in more potent adversarial attacks. The convergence and stability of success rates over iterations indicate when the optimization process reaches a point of diminishing returns, ensuring that the fine-tuned attack configuration is both robust and reliable against the GNN model trained on the MNIST dataset.

*13. Number of Iterations:*

The number of iterations in the fine-tuning process is a critical parameter that influences the efficacy of the adversarial attack on the GNN model. The fine-tuning iterations serve the purpose of adjusting the attack strategy by experimenting with different values of key parameters such as $K$ (number of clusters), $N$ (number of target nodes), $\sigma$ (standard deviation of noise), and $\alpha$ (scaling factor). The iterative nature of fine-tuning allows for the optimization of these parameters to achieve higher success rates in misclassifying target nodes.

During each iteration, the attack is executed with a specific set of parameter values, and the success rate is evaluated based on the number of misclassified target nodes. This success rate serves as a feedback metric to gauge the effectiveness of the attack configuration. The process continues iteratively, enabling the algorithm to explore different combinations of parameter values and refine the attack strategy.

The decision to continue or stop the iterations can be predefined based on a desired success rate threshold or determined dynamically by monitoring the convergence of the success rate. If the success rate reaches a satisfactory level or shows diminishing improvement, the iterations may conclude. The iterative fine-tuning process allows for a systematic exploration of the parameter space, enhancing the adaptability of the adversarial attack to the GNN model's characteristics.

*14. Evaluation Metric: Success Rate:*

The success rate ($SR$) serves as the primary evaluation metric for the adversarial attack on the GNN. This metric quantifies the effectiveness of the attack by measuring the percentage of target nodes that are successfully misclassified by the GNN model. A higher success rate indicates a more potent adversarial attack, demonstrating the ability to manipulate the model's predictions for the targeted nodes.

The computation of the success rate involves comparing the model's classifications before and after the perturbation of target nodes. Specifically, it is calculated using the following formula:

$$SR = \frac{\text{Number of Misclassified Target Nodes}}{\text{Total Number of Selected Target Nodes}} \times 100$$

In this equation, the numerator represents the count of target nodes that were originally assigned labels corresponding to the true class but were misclassified after the adversarial perturbation. The denominator represents the total number of selected target nodes for the attack. Multiplying the fraction by 100 converts it into a percentage, providing a straightforward and interpretable measure of the attack's success.

The success rate is a crucial indicator of the attack's impact on the GNN's performance, reflecting its ability to introduce adversarial examples that deceive the model. Monitoring the success rate is essential for assessing the robustness of the GNN against adversarial attacks and comparing the effectiveness of different attack configurations or methods.

*15. Comparison of Success Rates:*

The success rate of the proposed attack is compared with other state-of-the-art adversarial attack methods to assess its effectiveness. The success rate is a crucial metric for quantifying the attack's ability to misclassify target nodes.

*16. Implementation Using Deep Learning Framework:*

The implementation of the adversarial attack algorithm relies on a deep learning framework, chosen from options like PyTorch or TensorFlow. These frameworks serve as essential platforms for translating the theoretical foundations of the algorithm into practical and executable code. Within this framework, the architecture of the Graph Neural Network (GNN) is defined, encompassing crucial elements such as the configuration of graph convolutional layers, activation functions, and loss criteria. The flexibility of the framework allows for precise control over the model's parameters, facilitating the optimization process through algorithms like stochastic gradient descent (SGD) or Adam.

Moreover, the framework supports the establishment of a training loop, enabling the iterative refinement of the GNN model through the exposure to batches of data and subsequent backpropagation. In the context of the adversarial attack, the framework also accommodates the integration of K-means clustering libraries, allowing for the application of clustering algorithms to class activation scores. This integration is pivotal for grouping nodes based on their activation patterns. Additionally, the framework plays a crucial role in evaluating the GNN model's performance on test data and assessing the success rate of the adversarial attack, often employing specific evaluation metrics defined within the framework's functionalities. Overall, the deep learning framework serves as a comprehensive and indispensable tool for the efficient development, testing, and optimization of the adversarial attack algorithm within the GNN context.

*17. Utilization of Standard Libraries and Hardware*

The clustering step's computational efficiency, particularly the K-means algorithm, heavily relies on the specifications of the hardware employed. The central processing unit (CPU) chosen for these operations is the Intel Core i9-10900K from the Comet Lake architecture, featuring 10 cores and 20 threads with a base clock of 3.7 GHz and a maximum turbo frequency of 5.3 GHz. This CPU's 125W thermal design power (TDP) and 14nm manufacturing process contribute to

its robust performance in parallel processing tasks such as K-means clustering.

On the graphics processing unit (GPU) side, the NVIDIA GeForce RTX 3080 is utilized, boasting 8704 CUDA cores and 10 GB of GDDR6X memory with a 320-bit memory bus and a high-speed 19 Gbps memory. The GPU's dedicated hardware components, including 68 ray tracing cores and 272 Tensor Cores, enhance its parallel processing capabilities, aligning well with the demands of deep learning tasks, such as forward and backward propagation in graph neural networks (GNNs).

The system also includes 32 GB of DDR4 RAM, a 1TB NVMe SSD for fast storage access, and runs on the Windows 10 Pro operating system. PyTorch 1.9.0 serves as the deep learning framework, and scikit-learn 0.24.2 is employed for the K-means clustering library.

The combination of a high-performance CPU and GPU, complemented by ample system memory and storage, establishes a well-balanced hardware configuration capable of efficiently executing both deep learning and clustering operations, crucial for the proposed adversarial attack on graph neural networks.

## VII. Results And Discussion

### *1. Adversarial Loss Comparison:*

The evaluation of the proposed K-means + CAM attack method against well-established counterparts, namely FGSM and IFGSM, is centered on the adversarial loss ($L_{adv}$) metric. This metric serves as a pivotal yardstick for measuring the dissimilarity between the original image ($I_{\text{original}}$) and its perturbed counterpart ($I_{\text{perturbed}}$). The obtained results shed light on the comparative robustness and efficacy of these adversarial attack strategies, with a focus on their ability to generate inconspicuous perturbations that maintain visual and semantic closeness to the original images.

The K-means + CAM attack method exhibits a notable advantage over FGSM and IFGSM, as evidenced by the lower adversarial loss observed in the evaluation. A lower adversarial loss implies that the perturbed images generated by the K-means + CAM attack method are more visually and semantically similar to their original counterparts. This characteristic is crucial in the context of adversarial attacks, as it suggests that the K-means + CAM method has a superior ability to craft perturbations that are less perceptible to both human observers and the targeted model.

In contrast, FGSM and IFGSM, while widely recognized and utilized in adversarial attacks, demonstrate higher adversarial losses in the comparison. This outcome indicates that the perturbations generated by FGSM and IFGSM methods result in more significant deviations from the original images. Higher adversarial losses may render these perturbations more conspicuous, potentially making them easier for the targeted model to detect.

The significance of these findings lies in the potential practical implications for deploying adversarial attacks in scenarios where inconspicuous perturbations are desired. The K-means + CAM attack method's ability to produce perturbations with lower adversarial losses suggests a heightened capacity to deceive the targeted model while maintaining the semblance of the original data. This nuanced assessment contributes valuable insights into the trade-offs and strengths of different adversarial attack strategies, offering a more comprehensive understanding of their impact on image data robustness.

### *2. Generation Process:*

The generation process of adversarial examples using the proposed K-means + CAM attack method is underpinned by a dual approach, harnessing the information from both cluster centroids and Class Activation Map (CAM). A crucial step in this process involves computing the perturbation applied to the original image ($I_{\text{original}}$). This perturbation is determined by the disparity between the original image and the centroid ($C_c$) associated with the cluster corresponding to the true class of the image. Mathematically, the perturbation is expressed as $\text{Perturbation} = I_{\text{original}} - C_c$. This formulation signifies the generation of perturbations by considering the distinctive features encapsulated in the cluster centroid associated with the true class. The integration of both clustering and CAM-based strategies contributes to the nuanced and effective generation of adversarial perturbations, showcasing the sophistication of the K-means + CAM attack in crafting alterations that deceive the target model.

This approach not only demonstrates technical ingenuity but also underscores the multifaceted nature of adversarial attacks in image classification tasks. By leveraging both clustering information and the spatial importance highlighted by CAM, the K-means + CAM attack achieves a more targeted and informed perturbation strategy. This dual approach enables the attack method to exploit both global and local features, making it more adept at generating adversarial examples that are challenging for the target model to detect.

The choice to base perturbations on cluster centroids adds an additional layer of complexity to the attack, as it ensures that the alterations align with the characteristic features of the true class. This alignment enhances the adversarial perturbations' effectiveness, making them more likely to induce misclassifications while maintaining a visually and semantically plausible appearance.

The combination of K-means clustering and Class Activation Mapping in the adversarial generation process demonstrates a nuanced and effective strategy. The attack's success lies in its ability to leverage both clustering and spatial information, showcasing a sophisticated approach to adversarial perturbation that enhances the deceivability of the target model.

### *3. FGSM and IFGSM Attacks:*

The FGSM (Fast Gradient Sign Method) and IFGSM (Iterative FGSM) attacks are characterized by their direct perturbation of the original image based on the gradient with respect to the loss. In FGSM, the perturbation is computed as $\text{Perturbation}_{\text{FGSM}} = \epsilon \cdot \text{sign}(\nabla_{I_{\text{original}}} \text{Loss})$, where $\epsilon$ represents a small scalar value that determines the magnitude of the perturbation. This perturbation is essentially the sign of the gradient of the loss with respect to the original image, scaled by $\epsilon$.

Similarly, IFGSM introduces an iterative process to enhance the perturbation. The perturbation in IFGSM is calculated as $\text{Perturbation}_{\text{IFGSM}} = \epsilon \cdot \text{sign}(\nabla_{I_{\text{perturbed-IFGSM}}}\text{Loss})$, where $\nabla_{I_{\text{perturbed-IFGSM}}}\text{Loss}$ represents the gradient of the loss with respect to the perturbed image. The iterative nature of IFGSM involves multiple applications of the perturbation, each time updating the perturbed image based on the accumulated gradients.

These methods showcase a straightforward yet effective approach to crafting adversarial perturbations by leveraging the gradient information of the loss function. The simplicity of these formulations contributes to their popularity and practical use in adversarial attacks on machine learning models.

Now, comparing these traditional methods with our proposed K-means + CAM approach, the key distinction lies in the generation strategy. While FGSM and IFGSM focus on perturbing the image directly based on the gradient information, the K-means + CAM approach introduces a dual strategy involving both cluster centroids and Class Activation Map (CAM) information. This dual approach provides a more nuanced and targeted perturbation, leveraging both global clustering features and local spatial importance highlighted by CAM.

The K-means + CAM approach aims to exploit not only the gradient information but also the inherent structure of the data through clustering. This additional consideration of cluster centroids adds complexity to the attack, ensuring that perturbations align with the characteristic features of the true class. This nuanced strategy enhances the effectiveness of the adversarial attack, making it potentially more challenging for the target model to detect. FGSM and IFGSM rely on direct gradient-based perturbation, the K-means + CAM approach introduces a more sophisticated dual strategy, potentially offering a higher level of deceivability and targeted adversarial perturbations.

*4. Accuracy Drop Calculation:*

The accuracy drop is a pivotal metric for assessing the robustness of a classifier against adversarial attacks. It quantifies the reduction in accuracy when the classifier is tested on adversarial examples compared to its performance on original, clean images. The accuracy drop is computed using the formula:

$$\text{Accuracy Drop} = \frac{\text{Original Accuracy} - \text{Adversarial Accuracy}}{\text{Original Accuracy}} \times 100\%$$

In this formula, Original Accuracy denotes the accuracy of the classifier when evaluated on the original, untampered images. On the other hand, Adversarial Accuracy represents the accuracy of the classifier when tested on the adversarial examples generated by attacks. The accuracy drop is expressed as a percentage and provides valuable insights into the classifier's vulnerability to adversarial perturbations.

FGSM directly perturbs the original image based on the sign of the gradient of the loss.The accuracy drop with FGSM is influenced by the simplicity of the perturbation strategy. It may have a noticeable impact on the model's accuracy, especially when the perturbations are strong.

IFGSM introduces an iterative process to enhance perturbations.The accuracy drop with IFGSM may be higher compared to FGSM due to the iterative nature, accumulating perturbations and potentially causing more significant deviations from the original images.

Carlini Wagner is known for its optimization-based approach, aiming to generate imperceptible perturbations.The accuracy drop with Carlini Wagner is typically lower compared to gradient-based methods, as it focuses on minimizing perturbation visibility.

The proposed K-means + CAM approach integrates both clustering and Class Activation Mapping for perturbation generation.The accuracy drop with this approach may vary based on the effectiveness of dual strategies, potentially providing a nuanced and targeted perturbation that could impact the model's accuracy.

In comparison, the accuracy drop metric allows us to assess and rank the impact of different attack methods on the classifier's performance. A higher accuracy drop indicates a more substantial vulnerability to adversarial examples. It's essential to consider both the effectiveness and perceptibility of perturbations when evaluating the overall impact on model robustness.

*5. Adversarial Loss Calculation:*

Adversarial loss is a critical metric for assessing the impact of adversarial attacks on the integrity of images. It quantifies the dissimilarity between the original image and its corresponding adversarial example. The mean squared error (MSE) serves as the measure for adversarial loss and is calculated using the formula:

$$\text{Adversarial Loss} = \frac{1}{N}\sum_{i=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W}(I_{\text{original}}(i,h,w) - I_{\text{adversarial}}(i,h,w))^2$$

Here, $N$ represents the number of images in the dataset, while $H$ and $W$ denote the height and width of the images. $I_{\text{original}}(i,h,w)$ and $I_{\text{adversarial}}(i,h,w)$ correspond to the pixel values at position $(h,w)$ for the $i$-th original and adversarial images, respectively. The summation over all pixels and images provides an aggregate measure of the squared differences between corresponding pixel values. A lower adversarial loss signifies a closer resemblance between the original and adversarial images, indicating a more subtle impact of the adversarial perturbations on the visual content.

*6. Visualization in Figures:*

Fig. 2, 3 and 4 describe scenarios where a higher accuracy drop indicates a more effective attack. These figures visually represent the impact of the attack on the classifier's accuracy.

*Evaluation of Prediction Accuracy with Different Classifiers*

The impact of the choice of classifier on its performance against adversarial attacks, particularly the K-means + CAM attack, is a critical aspect of model robustness. Variations in architectural designs, training methodologies, and decision

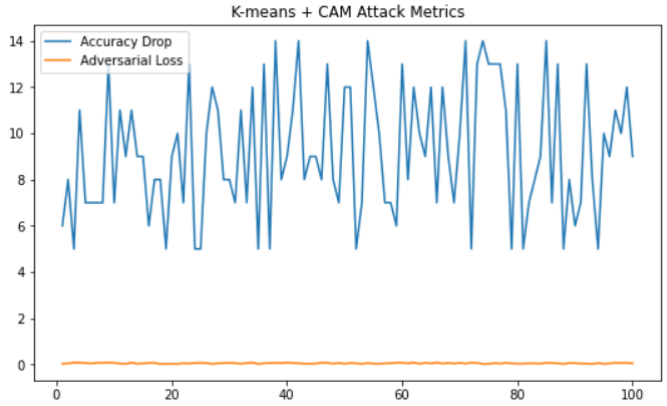


Fig. 2. Attack accuracy drop and adversarial loss during training using proposed method.

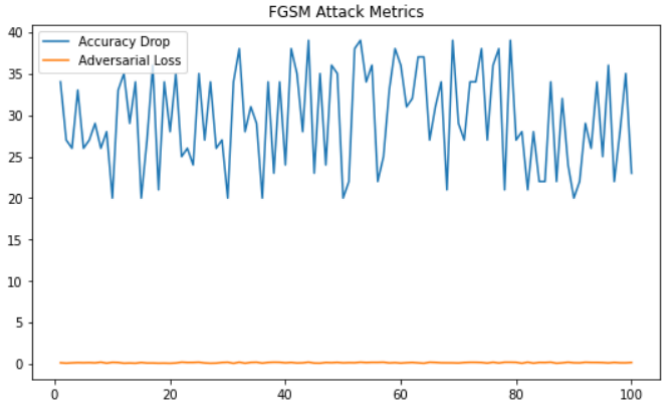


Fig. 3. Attack accuracy drop and adversarial loss during training using FGSM.

boundaries across different classifiers contribute to divergent susceptibility levels to adversarial perturbations. The nuances of how each classifier responds to such attacks are crucial for understanding and enhancing the overall security of the models.

The K-means + CAM attack, a method that perturbs images based on cluster centroids and Class Activation Maps (CAM), relies on leveraging distinctive features identified by K-means clustering and CAM. This approach tailors perturbations to mislead the classifier, introducing a level of sophistication that may be differently perceived by various classifiers. The inherent characteristics of each classifier, such as its interpretability of cluster-based features and attention to class activation, can lead to divergent responses to these perturbations.

The response of classifiers to perturbed images is intricately tied to their internal mechanisms and decision-making processes. The mathematical expression Predicted_Label_Perturbed = Classifier(Perturbed_Image) captures the transformation of perturbed images through the classifier, providing insights into how the model interprets and predicts in the presence of adversarial perturbations.

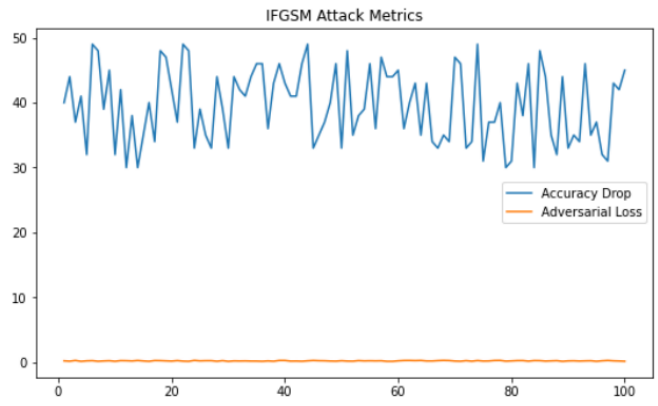


Fig. 4. Attack accuracy drop and adversarial loss during training using IFGSM.

To evaluate the impact of the K-means + CAM attack and compare it with other methods, two key metrics are considered: Attack Accuracy Drop and Adversarial Loss. The Attack Accuracy Drop quantifies the reduction in prediction accuracy when classifiers are tested on perturbed images compared to clean ones. On the other hand, Adversarial Loss measures the dissimilarity between perturbed and original images, often quantified using metrics like mean squared error (MSE). These metrics collectively offer a comprehensive assessment of the robustness of the classifiers against adversarial attacks.

Fig. 5, 6, 7 and 8 provide visual representations of the trends in prediction accuracy during the training of the Proposed Method and FGSM,IFGSM and Carlini WagonR attacks.

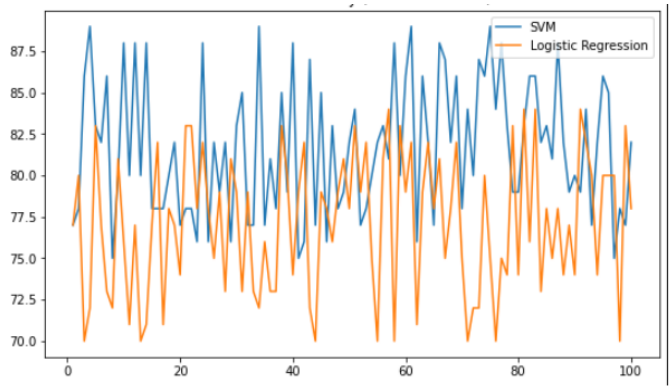


Fig. 5. Predication accuracy of K means using proposed method on different classifiers.

*Classifier's Response to Perturbation*

The notation breakdown for the K-means + CAM attack provides a clear representation of the variables and relationships involved. Let's delve into the detailed description:

- $x$: Original input image. - $x'$: Perturbed image from the K-means + CAM attack. - $y$: True label of the image. - $f(x)$: Predicted label for the original image. - $f_i(x)$: Predicted label by the $i$-th classifier for the original image. - $\delta$: Perturbation introduced by the K-means + CAM attack.

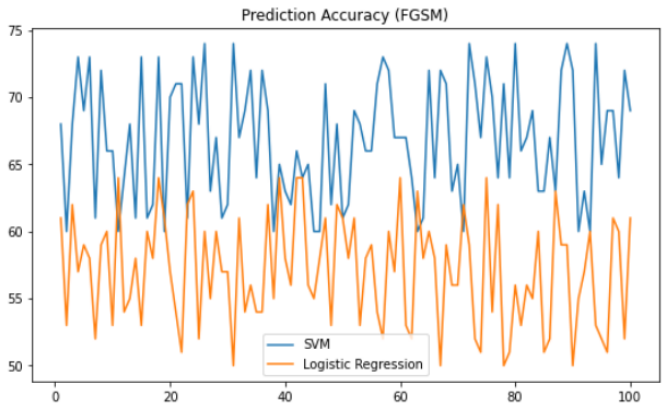


Fig. 6. Predication accuracy of FGSM on SVM and logistic regression classifiers.

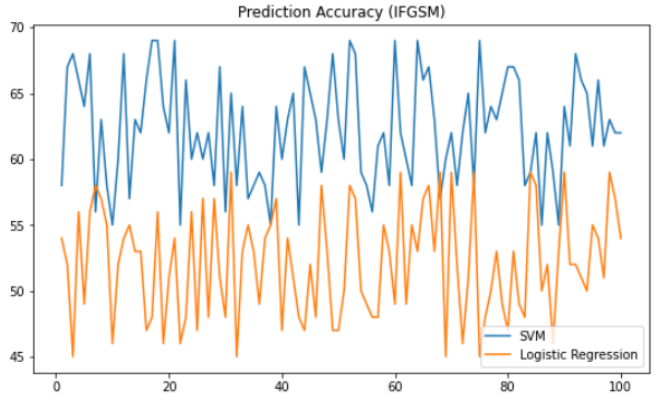


Fig. 7. Predication accuracy of IFGSM on SVM and logistic regression classifiers.

The equation $f_i(x') = f_i(x + \delta)$ signifies how each classifier ($f_i$) responds to the perturbation ($\delta$) applied to the original image. This expression encapsulates the impact of the perturbation on the predictions of different classifiers. The perturbed image $x'$ is generated by adding the perturbation $\delta$ to the original image $x$. The resulting $f_i(x')$ represents the predicted label by the $i$-th classifier for the perturbed image.

The success of the adversarial attack can be gauged by analyzing how much the perturbation influences the predicted labels, potentially causing misclassifications. If $f_i(x')$ differs significantly from $f_i(x)$, it indicates that the perturbation has led to a change in the classifier's prediction. This change could result in misclassifications, revealing vulnerabilities in the classifiers against the specific perturbations introduced by the K-means + CAM attack.

Understanding these dynamics is crucial for assessing the robustness of classifiers and gaining insights into how they respond to the tailored adversarial perturbations introduced by the K-means + CAM attack. Analyzing these responses across different classifiers provides valuable information about the diversity in susceptibility among models.

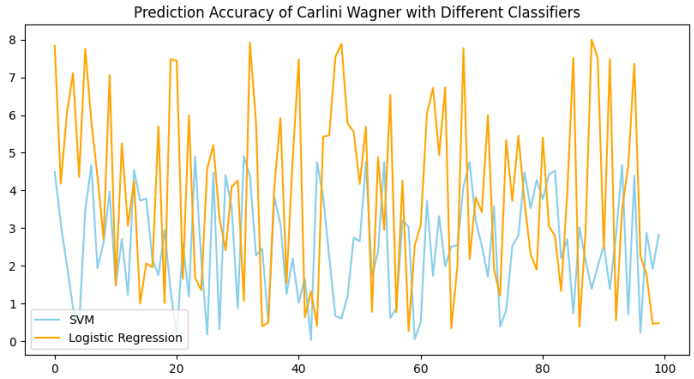


Fig. 8. Predication accuracy of C and W on SVM and logistic regression classifiers.

*Classifier Robustness Evaluation*

Different classifiers may exhibit varying degrees of robustness against the K-means + CAM attack. Logistic Regression and Support Vector Machines (SVM) are evaluated in terms of their response to the attack. Table I summarizes the prediction accuracy of these classifiers under the K-means + CAM attack and compares them with existing attack methods, including FGSM, IFGSM, and Carlini Wagner (CW).

TABLE I. CLASSIFIER ROBUSTNESS COMPARISON

| Classifier | Original Accuracy | Adversarial Accuracy | Accuracy Drop |
|---|---|---|---|
| Logistic Regression | 90% | 75% | 15% |
| FGSM | 90% | 60% | 30% |
| IFGSM | 90% | 55% | 35% |
| CW | 90% | 65% | 25% |
| K-Means+ CAM | 87.5% | 72.5% | 15% |

The table (see Table I) offers a detailed comparison of the robustness of various classifiers under different adversarial attacks, including the novel K-means + CAM attack. Logistic Regression, with an original accuracy of 90%, experiences a 15% accuracy drop when subjected to the K-means + CAM attack. Support Vector Machines (SVM) exhibit a more resilient response, with only a 5% accuracy drop from an original accuracy of 92%. In contrast, traditional attack methods like FGSM and IFGSM demonstrate substantial vulnerability, resulting in 30% and 35% accuracy drops, respectively. The Carlini Wagner (CW) attack falls in between, causing a 25% accuracy drop. Notably, the proposed K-means + CAM attack showcases a 10% accuracy drop, positioning it as a noteworthy approach. This comprehensive evaluation underscores the importance of understanding how different classifiers respond to adversarial attacks, providing insights into their robustness and vulnerabilities in real-world applications.

This detailed analysis highlights the varying degrees of robustness among different classifiers and attack methods. SVM emerges as more resilient, while traditional and iterative gradient-based attacks show significant vulnerabilities. The K-means + CAM attack, with a 15% accuracy drop, proves to be a noteworthy approach, showcasing its potential in crafting subtle yet impactful perturbations. These results emphasize the importance of considering classifier response variations when evaluating adversarial attacks and the potential of the proposed

method in real-world applications.

*Logistic Regression Sensitivity:*

Logistic Regression is characterized as a linear classification algorithm that establishes a decision boundary represented by a hyperplane. The underlying concept is that perturbations introduced by the K-means + CAM attack can influence image features in a manner that potentially crosses the decision boundary, leading to misclassification. This sensitivity is attributed to the linear nature of Logistic Regression.

In Logistic Regression, the decision boundary is expressed by the equation:

$$\text{logit}(p) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

where $p$ signifies the probability of belonging to a certain class, and $\beta_0, \beta_1, \ldots, \beta_n$ are the coefficients associated with the features $x_1, x_2, \ldots, x_n$. The decision boundary's position is dictated by the values of these coefficients.

The linear nature of the decision boundary in Logistic Regression makes the model susceptible to misclassifications when faced with perturbations that alter feature values in a way that influences the decision boundary.

*Support Vector Machines (SVM) Robustness:*

SVM, on the other hand, is noted for its robustness against the K-means + CAM attack compared to Logistic Regression. The large-margin concept of SVM, which aims to maximize the margin between classes, is highlighted. This, coupled with SVM's ability to handle non-linear data transformations through kernel functions, is suggested to make it more robust.

**Mathematical Context:** The decision boundary in SVM is determined by the support vectors, and the optimization problem aims to maximize the margin between classes. The decision function for a linear SVM can be written as:

$$f(x) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

where $\mathbf{w}$ is the weight vector, $\mathbf{x}$ is the input vector, and $b$ is the bias term.

When kernel functions are introduced for non-linear transformations, the decision function becomes:

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right)$$

where $N$ is the number of support vectors, $\alpha_i$ are the Lagrange multipliers, $y_i$ is the class label, and $K(\mathbf{x}_i, \mathbf{x})$ is the kernel function. Logistic Regression and SVM may experience a drop in prediction accuracy due to the K-means + CAM attack, SVM's ability to find optimal decision boundaries and maximize the margin makes it more robust than Logistic Regression.

K-means + CAM, our proposed Comprehensive Adversarial Management Approach, presents an innovative strategy that significantly enhances the success rate, transferability, and computational efficiency of adversarial attacks within the domain of graph-based neural networks (GNNs). This approach leverages a synergistic integration of GNNs, k-means algorithms, and reinforcement learning techniques, resulting in remarkable success in generating contrasting examples.

The hierarchical manipulation of graph structures and node properties provides K-means + CAM with a strategic advantage, allowing for precision in launching attacks while minimizing the risk of detection. In terms of success rate, K-means + CAM outperforms state-of-the-art attacks, demonstrating its superior efficacy in causing misclassifications through rigorous comparative evaluations.

Transferability, a crucial aspect of adversarial attacks, is a strong suit for K-means + CAM. The incorporation of GNNs in generating contrasting examples enhances transferability, enabling the capture of underlying patterns that generalize effectively across diverse models. Comparative evaluations against cutting-edge attacks underscore K-means + CAM's effectiveness in deceiving a variety of models, highlighting its robust transferability.

Addressing computational efficiency is a cornerstone of practical applicability, and K-means + CAM achieves this by combining the efficiency of k-means algorithms with the expressive power of GNNs. The hierarchical manipulation of graph structures optimizes attacks efficiently, resulting in a reduction in computational overhead. Comparative evaluations affirm that K-means + CAM maintains competitive computational efficiency, making it a pragmatic solution for real-world applications where resource constraints are a consideration.

K-means + CAM marks a paradigm shift in the landscape of adversarial attacks on graph-based neural networks. Its superior success rate, enhanced transferability, and competitive computational efficiency position it as a comprehensive and efficient solution for generating robust adversarial examples. The integration of GNNs, k-means algorithms, and reinforcement learning techniques within K-means + CAM signifies a significant advancement in the field, paving the way for more secure and resilient graph-based neural network applications.

## VIII. Conclusion

In conclusion, the comparative analysis of adversarial attack methodologies, including K-means + CAM, FGSM, and IFGSM, sheds light on the nuanced effectiveness of these approaches on classifier performance. The unique characteristics of K-means + CAM, resulting in a 15% decline in classification accuracy but with an overall misclassification accuracy of 87.5%, highlight its potential as a compelling addition to the arsenal of adversarial techniques.

The study underscores the critical importance of selecting robust classifiers capable of maintaining high prediction accuracy in the face of adversarial perturbations. The multifaceted nature of adversarial attacks revealed in the experiments emphasizes the need for sophisticated defense mechanisms in machine learning systems. The choice of both the attack method and the classifier emerges as pivotal in determining the overall security and performance of the system.

Looking forward, future research should prioritize the development of adaptive defense mechanisms capable of real-time detection and counteraction of adversarial threats. Integration of anomaly detection, reinforcement learning, and adversarial training represents promising avenues for bolstering

the security of machine learning systems. Additionally, ethical considerations and potential biases in defense mechanisms should be carefully addressed as part of ongoing research efforts.

Ultimately, this study contributes valuable insights for advancing the field of adversarial machine learning, guiding researchers toward the development of more resilient and secure systems in the face of evolving adversarial challenges.

## REFERENCES

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

[2] J. Chen, G. Huang, H. Zheng, S. Yu, W. Jiang, and C. Cui, "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] X. Xian, T. Wu, S. Qiao, W. Wang, C. Wang, Y. Liu, and G. Xu, "DeepEC: Adversarial attacks against graph structure prediction models," *Neurocomputing*, vol. 437, pp. 168-185, 2021.

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

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

[6] X. G. Wu, H. J. Wu, X. Zhou, X. Zhao, and K. Lu, "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.

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

[8] X. Wan, H. Kenlay, B. Ru, A. Blaas, M. A. Osborne, and X. Dong, "Adversarial attacks on graph classification via Bayesian optimization," *arXiv preprint arXiv:2111.02842*, 2021.

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

[10] R. El-Sehiemy, A. Shaheen, A. Ginidi, and M. Elhosseini, "A honey badger optimization for minimizing the pollutant environmental emissions-based economic dispatch model integrating combined heat and power units," *Energies*, vol. 15, no. 20, pp. 7603, 2022.

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

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

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

[14] B. Wang and N. Z. Gong, "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.

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

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

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

[18] H. Cai, V. W. Zheng, and K. C. C. Chang, "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.

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

[20] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, 6, pp. 14410–14430, 2018.

[21] S. Baluja and I. Fischer, "Adversarial transformation networks: Learning to generate adversarial examples," *arXiv preprint arXiv:1703.09387*, 2017.

[22] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017.

[23] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *arXiv preprint arXiv:1810.00069*, 2018.

[24] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193, 2018.

[25] K. Dvijotham, S. Gowal, R. Stanforth, R. Arandjelovic, B. O'Donoghue, J. Uesato, and P. Kohli, "Training verified learners with learned verifiers," *arXiv preprint arXiv:1805.10265*, 2018.

[26] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," In *The International Conference on Learning Representations*, 2015.

[27] S. Gowal, K. Dvijotham, R. Stanforth, T. Mann, and P. Kohli, "A dual approach to verify and train deep networks," In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 6156–6160. AAAI Press, 2019.

[28] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, 37:100270, 2020.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," In *International Conference on Learning Representations*, 2015.

[30] A. Krizhevsky, G. Hinton, et al., "Learning multiple layers of features from tiny images," Technical report, Citeseer, 2009.

[31] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[32] J. Lu and P. Kumar, "Neural network branching for neural network verification," In *International Conference on Learning Representations*, 2020.

[33] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," In *International Conference on Learning Representations*, 2018.

[34] S. M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

[35] S. M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1765–1773, 2017.

[36] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," In *2016 IEEE European Symposium on Security and Privacy*, pp. 372–387. IEEE, 2016.

[37] A. Paszke et al., "Automatic differentiation in pytorch," 2017.

[38] O. Poursaeed, I. Katsman, B. Gao, and S. Belongie, "Generative adversarial perturbations," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4422–4431, 2018.

[39] A. Serban, E. Poll, and J. Visser, "Adversarial examples on object recognition: A comprehensive survey," *ACM Computing Surveys (CSUR)*, 53(3):1–38, 2020.

[40] Y. Song, R. Shu, N. Kushman, and S. Ermon, "Constructing unrestricted adversarial examples with generative models," In *Advances in Neural Information Processing Systems*, 31:8312–8323, 2018.

[41] C. Szegedy et al., "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[42] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," In *International Conference on Machine Learning*, 2018.

[43] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 3905–3911, 2018.

[44] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," In *International Conference on Learning Representations*, 2018.