

In this section, the **multi-head attention mechanism** is introduced to stabilize the learning process. Here's a breakdown:

1. **Normalized Attention Coefficients:** After computing the attention coefficients  $\alpha_{ij}$  (using the attention mechanism), they are **normalized** across all neighbors for each node. These coefficients are used to weight the contribution of each neighboring node to the target node's representation.
2. **Linear Combinations:** The normalized attention coefficients are multiplied by the corresponding input node features (the representations of neighboring nodes) and summed to create a **linear combination**. This combination is then passed through a **nonlinear activation function** (such as ReLU) to generate the output node representation.
3. **Multi-head Attention:** Instead of relying on a single attention mechanism, the model uses **multiple attention heads** (denoted by K). Each attention head learns an independent attention aggregation. This helps the model to capture different aspects of the node relationships, making the learning process more stable and effective.
4. **Output Representations:** The output of each attention head is calculated for both the **positive GNN** and the **negative GNN**:

Message ChatGPT



4. **Output Representations:** The output of each attention head is calculated for both the **positive GNN** and the **negative GNN**:

- For **positive GNN**:  $\mathbf{h}_i^+$  is the output representation for node  $v_i$ , considering positive neighbors.
- For **negative GNN**:  $\mathbf{h}_i^-$  is the output representation for node  $v_i$ , considering negative neighbors.

5. **Final Node Representation:** After applying multi-head attention across multiple layers, the model **concatenates** the representations from all layers to form the final node representation. This allows the model to capture connectivity at different levels (or orders) of the graph structure.

So, to summarize:

- $\alpha_{ij}^k$ : Attention coefficient for the **k-th** attention head, between node  $v_i$  and its neighbor  $v_j$ .
- $W_b^+, W_b^-$ : Transformation matrices used to adjust the node features in both the positive and negative GNNs.
- **Multiple heads** allow the model to learn multiple ways to aggregate neighbor information, leading to more robust node embedding ↓

Message ChatGPT



view  $\tilde{\mathcal{G}}_i = (\mathcal{V}, \tilde{\mathcal{E}}_i^+, \tilde{\mathcal{E}}_i^-)$  into two graphs containing only positive links and only negative links, referred to as **positive graph**  $\tilde{\mathcal{G}}_i^+ = (\mathcal{V}, \tilde{\mathcal{E}}_i^+)$  and **negative graph**  $\tilde{\mathcal{G}}_i^- = (\mathcal{V}, \tilde{\mathcal{E}}_i^-)$  respectively. A **positive GNN** is leveraged to learn node representations from the positive graphs while a **negative GNN** is employed to learn node representations from the negative graphs, which are shared across all views. Positive and negative node representations from positive graphs are denoted as **consistent representations** while the ones from negative graphs are denoted as **inconsistent representations**.

In the following, we introduce the aggregation layer used in both positive and negative GNNs in detail. Since the contributions of neighbors are different for each node, we utilize the attention mechanism to learn neighbors' importance for each node during the aggregation process. Concretely, for a given node  $v_i$  and its neighbors in the positive/negative graph, we leverage the attentional mechanism parametrized by a weight vector  $\mathbf{a}^+ \in \mathbb{R}^{1 \times 2d_{out}}$  for the positive GNN (or  $\mathbf{a}^- \in \mathbb{R}^{1 \times 2d_{out}}$  for the negative GNN) and apply LeakyReLU nonlinearity (with negative input slope  $\alpha = 0.2$ ) to calculate the attention coefficient  $\alpha_{ij}$  of node  $v_j$  to node  $v_i$ , which is formulated as follows:

$$\frac{\exp(\text{LeakyReLU}(\mathbf{a}^+ [\mathbf{h}_i^+ \mathbf{W}_a^+ || \mathbf{h}_j^+ \mathbf{W}_a^+]^T))}{\sum_{t \in \mathcal{N}_i^+} \exp(\text{LeakyReLU}(\mathbf{a}^+ [\mathbf{h}_i^+ \mathbf{W}_a^+ || \mathbf{h}_t^+ \mathbf{W}_a^+]^T))}, \quad (1)$$

$$\frac{\exp(\text{LeakyReLU}(\mathbf{a}^- [\mathbf{h}_i^- \mathbf{W}_a^- || \mathbf{h}_j^- \mathbf{W}_a^-]^T))}{\sum_{t \in \mathcal{N}_i^-} \exp(\text{LeakyReLU}(\mathbf{a}^- [\mathbf{h}_i^- \mathbf{W}_a^- || \mathbf{h}_t^- \mathbf{W}_a^-]^T))}, \quad (2)$$

and  $\mathcal{N}_i^+$  and  $\mathcal{N}_i^-$  denote neighbors of node  $v_i$  in the positive and negative graph respectively,  $\mathbf{h}_i^+, \mathbf{h}_i^- \in \mathbb{R}^{d_{in}}$  represent consistent and inconsistent representation of node  $v_i$ ,

$\mathbf{W}_a^+, \mathbf{W}_a^- \in \mathbb{R}^{d_{in} \times d_{out}}$  are the weight matrices in the positive and negative GNNs respectively,  $d_{in}$  and  $d_{out}$  denote the input and output representation dimensions of the aggregation layer and we set  $d_{out}$  to be equal to  $d$  in each aggregation layer. In addition,  $\cdot^T$  denotes transposition and  $\parallel$  represents concatenation. Obviously, the attention coefficient is normalized across all choices of node  $v_j$ .

After obtaining the normalized attention coefficients, we compute the linear combinations of input node representations corresponding to the coefficients and apply a nonlinearity to generate the output representations for each node. Furthermore, the multi-head attention mechanism [37] which learns  $K$  independent attention aggregations are leveraged to make the learning process more stable, i.e.,

$$\mathbf{h}_i^{+'} = \left\| \sigma \left( \sum_{k=1}^K \alpha_{ij}^{k,+} \mathbf{h}_j^+ \mathbf{W}_b^{k,+} \right), \quad (3)$$

$$\mathbf{h}_i^{-'} = \left\| \sigma \left( \sum_{k=1}^K \alpha_{ij}^{k,-} \mathbf{h}_j^- \mathbf{W}_b^{k,-} \right), \quad (4)$$

where  $\mathbf{h}_i^{+'}, \mathbf{h}_i^{-'} \in \mathbb{R}^{d_{out}}$  represent the output consistent and inconsistent representations respectively,  $\alpha_{ij}^{k,+}, \alpha_{ij}^{k,-}$  and  $\mathbf{W}_b^{k,+}, \mathbf{W}_b^{k,-}$  are the normalized attention coefficients and the transformation matrices for positive and negative GNNs in the  $k$ -th head,  $\sigma$  denotes the nonlinear activation function such as ReLU.

Ultimately, the positive and negative GNNs are built by stacking multiple aggregation layers proposed in Eq. 3- 4, which are shared across all graph views, as shown in Fig. 3. Since representations in different layers emphasize the connectivity of different order, we concatenate representations of all layers to constitute the final

view and intra-view consistency. The inter-view consistency maximizes the agreement between node in different graphs. The similarities between the learned representations in the positive graph and the negative graph are measured by the cosine similarity between the learned representations. The contrastive learning loss between the learned representations in the positive graph and the negative graph is calculated as follows:

In the following, we introduce the objective for consistency learning. Let a batch  $\mathcal{B}$  containing  $m$  samples,  $\mathbf{z}_{i,m}^+$  and  $\mathbf{z}_{i,m}^-$  be the inter-view positive and negative samples generated from the  $i$ -th node in the  $m$ -th augmented graph. The inter-view consistency loss is calculated as follows:

$$\mathcal{L}_{inter\_pos} = -\log \tau(\mathbf{z}_{i,m}^+ | \mathbf{z}_{i,m}^-)$$

where  $\mathbf{z}_{i,m}^+$  represents the  $i$ -th node's representation in the  $m$ -th augmented graph and  $\tau$  denotes the cosine similarity function.

Similarly, for a node  $v_i$  in view  $i$ , the intra-view positive samples are generated from the same node in other graph views. The intra-view consistency loss is calculated as follows:

each graph view  $\mathcal{G}_i = (\mathcal{V}, \mathcal{E}_i^+, \mathcal{E}_i^-)$  into two graphs containing only positive links and only negative links, referred to as **positive graph**  $\tilde{\mathcal{G}}_i^+ = (\mathcal{V}, \tilde{\mathcal{E}}_i^+)$  and **negative graph**  $\tilde{\mathcal{G}}_i^- = (\mathcal{V}, \tilde{\mathcal{E}}_i^-)$  respectively. Then a **positive GNN** is leveraged to learn node representations from the positive graphs while a **negative GNN** is employed to learn representations from the negative graphs, which are shared across the positive and negative graphs of all views, respectively. The learned representations from positive graphs are denoted as **consistent representations** while the ones from negative graphs are denoted as **inconsistent representations**.

In the following, we introduce the aggregation layer used in positive and negative GNNs in detail. Since the contributions of neighbors are different for each node, we utilize the attention mechanism to learn neighbors' importance for each node during the aggregation process. Concretely, for a given node  $v_i$  and its neighbor  $v_j$  in the positive/negative graph, we leverage the attentional mechanism parametrized by a weight vector  $\mathbf{a}^+ \in \mathbb{R}^{1 \times 2d_{out}}$  for the positive GNN (or  $\mathbf{a}^- \in \mathbb{R}^{1 \times 2d_{out}}$  for the negative GNN) and apply the LeakyReLU nonlinearity (with negative input slope  $\alpha = 0.2$ ) to compute the attention coefficient  $\alpha_{ij}$  of node  $v_j$  to node  $v_i$ , which can be formulated as follows:

$$\alpha_{ij}^+ = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^+ [\mathbf{h}_i^+ \mathbf{W}_a^+ || \mathbf{h}_j^+ \mathbf{W}_a^+]^T))}{\sum_{t \in N_i^+} \exp(\text{LeakyReLU}(\mathbf{a}^+ [\mathbf{h}_i^+ \mathbf{W}_a^+ || \mathbf{h}_t^+ \mathbf{W}_a^+]^T))}, \quad (1)$$

$$\alpha_{ij}^- = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^- [\mathbf{h}_i^- \mathbf{W}_a^- || \mathbf{h}_j^- \mathbf{W}_a^-]^T))}{\sum_{t \in N_i^-} \exp(\text{LeakyReLU}(\mathbf{a}^- [\mathbf{h}_i^- \mathbf{W}_a^- || \mathbf{h}_t^- \mathbf{W}_a^-]^T))}, \quad (2)$$

where  $N_i^+$  and  $N_i^-$  denote neighbors of node  $v_i$  in the positive graph and negative graph respectively,  $\mathbf{h}_i^+, \mathbf{h}_i^- \in \mathbb{R}^{d_{in}}$  represent the input consistent and inconsistent representation of node  $v_i$ ,

$\mathbf{W}_a, \mathbf{W}_a^- \in \mathbb{R}^{d_{in} \times d_{out}}$  are the weight matrices in the positive and negative GNNs respectively,  $d_{in}$  and  $d_{out}$  denote the input and output representation dimensions of the aggregation layer and we set  $d_{out}$  to be equal to  $d$  in each aggregation layer. In addition,  $\cdot^T$  denotes transposition and  $||$  represents concatenation. Obviously, the attention coefficient is normalized across all choices of node  $v_j$ .

After obtaining the normalized attention coefficients, we compute the linear combinations of input node representations corresponding to the coefficients and apply a nonlinearity to generate the output representations for each node. Furthermore, the multi-head attention mechanism [37] which learns  $K$  independent attention aggregations are leveraged to make the learning process more stable, i.e.,

$$\mathbf{h}_i^{+'} = \left\| \sigma \left( \sum_{k=1}^K \sum_{j \in N_i^+} \alpha_{ij}^{k,+} \mathbf{h}_j^+ \mathbf{W}_b^{k,+} \right) \right\|, \quad (3)$$

$$\mathbf{h}_i^{-'} = \left\| \sigma \left( \sum_{k=1}^K \sum_{j \in N_i^-} \alpha_{ij}^{k,-} \mathbf{h}_j^- \mathbf{W}_b^{k,-} \right) \right\|, \quad (4)$$

where  $\mathbf{h}_i^{+'}, \mathbf{h}_i^{-'} \in \mathbb{R}^{d_{out}}$  represent the output consistent and inconsistent representations respectively,  $\alpha_{ij}^{k,+}, \alpha_{ij}^{k,-}$  and  $\mathbf{W}_b^{k,+}, \mathbf{W}_b^{k,-}$  are the normalized attention coefficients and the transformation matrices for positive and negative GNNs in the  $k$ -th head,  $\sigma$  denotes the nonlinear activation function such as ReLU.

Ultimately, the positive and negative GNNs are built by stacking multiple aggregation layers proposed in Eq. 3- 4, which are shared across all graph views, as shown in Fig. 3. Since representations in different layers emphasize the connectivity of different order, we concatenate representations of all layers to constitute the final

now instead of one aggregation layer we are having k layers



Not exactly. Instead of having **K aggregation layers**, we have **K attention heads in a single aggregation layer**. Here's the difference:

### 1. Multi-Head Attention in a Single Layer

- Instead of relying on a **single** attention mechanism to determine neighbor importance, we use **K independent attention heads**.
- Each head computes its own attention coefficients and applies them to aggregate neighbor features separately.
- The outputs from **all K heads are concatenated or averaged** to produce the final output representation for that layer.

### 2. Stacking Multiple Aggregation Layers

- The model **stacks multiple aggregation layers** (each with multi-head attention).
- Each aggregation layer refines the node representations further, capturing higher-order relationships in the graph.



Message ChatGPT







### 3. Challenges in Signed Graph Representation

- Researchers are trying to learn **low-dimensional node representations** (embeddings) for **signed graphs** to use in tasks like **link prediction** and **network analysis**.
- **Problem:** Real-world signed graphs contain **interaction noise** (inaccurate or random interactions).
  - Example: A user may **report another user unfairly**, leading to **incorrect negative links**.
  - Existing methods use these noisy interactions, which can cause:
    - **Overfitting** (the model learns noise instead of patterns).
    - **Poor generalization** (bad performance on new data).



Message ChatGPT





- Embedding transforms each node into a **fixed-size vector** that can be used in **machine learning models**.

“

## 2. How Do Embeddings Work in Signed Graphs?

- In **signed graphs**, nodes are connected by **positive (+)** and **negative (-)** edges, which indicate **trust/distrust**, **friend/enemy**, or **like/dislike** relationships.
- A **good embedding** should encode:
  - Who are the node's friends? (positive edges)
  - Who are the node's enemies? (negative edges)
  - How close or far should two nodes be?

Example:



- Suppose we embed users in a **social trust network** (e.g., Bitcoin-OTC).

Message ChatGPT



- A **good embedding** should encode:
  - Who are the node's friends? (positive edges)
  - Who are the node's enemies? (negative edges)
  - How close or far should two nodes be?

Example:

- Suppose we embed users in a **social trust network** (e.g., Bitcoin-OTC).
- If **User A** trusts **User B** but distrusts **User C**, their embeddings should reflect:
  - A and B are close in vector space (similar embeddings).
  - A and C are far apart (dissimilar embeddings).

### 3. How Are Embeddings Generated

Different methods can be used to generate embeddings:

Message ChatGPT





“

### 3. How Are Embeddings Generated?

Different methods can be used to generate embeddings:

1. **Graph Neural Networks (GNNs)** → Use deep learning to learn node representations.
2. **Random Walks + Word2Vec (e.g., DeepWalk, node2vec)** → Treat graph connections like sentences and learn embeddings like word embeddings.
3. **Matrix Factorization (e.g., SVD, Laplacian Eigenmaps)** → Factorize an adjacency matrix to extract node features.
4. **Contrastive Learning (e.g., SGCL in the paper)** → Train a model to make similar nodes closer and dissimilar nodes farther in the embedding space.



Message ChatGPT





“

## 1. Problem with Existing Contrastive Learning in Graphs

- **Graph Contrastive Learning (GCL)** is an approach that helps models learn better representations by ensuring that **similar nodes** are **closer** and **dissimilar nodes** are **farther apart**.
- However, current GCL methods are designed for **unsigned graphs** (graphs with only positive edges).
- **Signed graphs** (which have both positive and negative edges) introduce new challenges due to their **complex structure and distinct semantics** (e.g., trust/distrust, friend/enemy).

## 2. Solution: Using Contrastive Learning for Signed Graphs

- The authors propose applying **contrastive learning** with graph augmentations to signed graphs.

Message ChatGPT





“

## 2. Solution: Using Contrastive Learning for Signed Graphs

- The authors propose applying **contrastive learning with graph augmentations** to signed graphs.
- **Graph augmentation** means modifying the graph structure (e.g., adding/removing edges or flipping edge signs) to help the model learn better representations.
- The goal is to make embeddings **more robust** (less sensitive to noise) and **generalized** (better performance on unseen data).

## 3. How Does This Work?

- **Positive Links ("Friends")** → Nodes should be **closer** in embedding space.

Message ChatGPT





ChatGPT

Share

D

challenges of contrastive learning on signed graphs. Here's a breakdown of how the solution addresses these challenges:

## 1. Redesigning Graph Augmentations for Signed Graphs:

- **Challenge:** Traditional graph augmentations like node dropping or attribute masking, used for unsigned graphs, don't consider the **positive** and **negative** relationships in signed graphs.
- **Solution:** SGCL introduces **two types of graph augmentations** tailored for signed graphs. These augmentations take into account the **complex relationships** between positive (friend) and negative (enemy) edges. By using these domain-specific augmentations, SGCL ensures that the graph structure and edge signs are correctly reflected in the learned representations.
- **Balance Theory:** SGCL incorporates **balance theory** to guide these augmentations. This theory suggests that in signed networks, positive and negative links should respect certain balance properties (e.g., the "friend-of-a-friend" rule). These principles guide the perturbations applied to the graph.

## 2. Combining Two Forms of Contrastive Learning in One Model:

- **Challenge:** Most graph contrastive learning methods focus on ensuring that augmented

Message ChatGPT





ChatGPT

Share

D

unsigned graphs, don't consider the **positive** and **negative** relationships in signed graphs.

- **Solution:** SGCL introduces **two types of graph augmentations** tailored for signed graphs. These augmentations take into account the **complex relationships** between positive (friend) and negative (enemy) edges. By using these domain-specific augmentations, SGCL ensures that the graph structure and edge signs are correctly reflected in the learned representations.
  - **Balance Theory:** SGCL incorporates **balance theory** to guide these augmentations. This theory suggests that in signed networks, positive and negative links should respect certain balance properties (e.g., the "friend-of-a-friend" rule). These principles guide the perturbations applied to the graph.

## 2. Combining Two Forms of Contrast in One Model:

- **Challenge:** Most graph contrastive learning methods focus on ensuring that **augmented representations** of the same node are close while pushing away those of different nodes. However, signed graphs require an **additional contrast**—nodes should be close to those connected by positive edges and far from those connected by negative edges.
- **Solution:** SGCL integrates **two types of contrastive learning**:

Message ChatGPT





ChatGPT

Share

D

## 2. Overcoming Two Forms of Contrast in One Model...

- **Challenge:** Most graph contrastive learning methods focus on ensuring that **augmented representations** of the same node are close while pushing away those of different nodes. However, signed graphs require an **additional contrast**—nodes should be close to those connected by positive edges and far from those connected by negative edges.
- **Solution:** SGCL integrates **two types of contrastive learning**:
  - **Inter-view Contrastive Learning:** Ensures that the node representations in different graph augmentations (views) are consistent with each other.
  - **Intra-view Contrastive Learning:** Ensures that for any single graph view, nodes connected by positive edges (friends) should have **similar embeddings**, while those connected by negative edges (enemies) should have **dissimilar embeddings**.
- This combination of intra-view and inter-view contrasts helps the model learn both the overall consistency between graph augmentations and the **structural balance** between positive and negative links.

## 3. Learning Representations with GNNs and Attention Mechanisms:

Message ChatGPT





ChatGPT

Share

D

( “ )

### 3. Learning Representations with GNNs and Attention Mechanisms:

- **GNNs:** SGCL uses **two graph neural networks (GNNs)** to process the signed graph and generate representations:
  - One GNN learns representations for **positive relationships (friends)**, capturing the notion of **affinity or similarity**.
  - The other GNN learns representations for **negative relationships (enemies)**, focusing on **dissimilarity**.
- **Attention Mechanisms:** These GNNs are enhanced with **attention mechanisms** to weigh the importance of each neighbor's influence in the final node embedding. This allows the model to focus on more relevant nodes, especially in the context of signed edges.

### 4. Combining Supervised Labels with Contrastive Objectives:

- SGCL integrates **supervised learning** (from labeled data) with **contrastive objectives** to optimize the model. The **supervised lab** ↓ provide additional information to guide the learning process, especially when labeled data for specific tasks (like link prediction or node

Message ChatGPT





ChatGPT

dissimilarity.

- **Attention Mechanisms:** These GNNs are enhanced with **attention mechanisms** to weigh the importance of each neighbor's influence in the final node embedding. This allows the model to focus on more relevant nodes, especially in the context of signed edges.

”

#### 4. Combining Supervised Labels with Contrastive Objectives:

- SGCL integrates **supervised learning** (from labeled data) with **contrastive objectives** to optimize the model. The **supervised labels** provide additional information to guide the learning process, especially when labeled data for specific tasks (like link prediction or node classification) is available.
- The **combined contrastive objective** balances both the node-level contrastive learning (based on the signed structure) and the supervised task objectives (based on label information).

#### Key Points in SGCL:

- **Graph Augmentations:** Specifically designed for signed graphs, taking into account the positive and negative edges.

Message ChatGPT





ChatGPT

Share

D

learning to help the model learn robust and generalizable embeddings, which are critical for performing downstream tasks like node classification, link prediction, and community detection.

first of all we create different views of a graph(using over scgl) now we feed these to gnn is this correct



Yes, you're on the right track! Here's the sequence of steps broken down:

### 1. Create Different Views of the Graph (using augmentations like in SGCL):

- First, you apply **graph augmentations** (such as node dropout, edge removal, edge flipping, etc.) to generate **multiple augmented versions of the graph**. These versions are designed to capture the graph's **structural diversity** while maintaining key characteristics relevant for learning.
- In **signed graphs**, augmentations are specially designed to account for the **positive and negative edges**, respecting the signed nature of relationships (e.g., friendship vs. enmity).



Message ChatGPT





world graph data.

our main agenda currently is to generate robust embeddings



Yes, generating **robust embeddings** is indeed the core goal, especially in the context of signed graphs and contrastive learning. Here's a concise overview of how this objective fits into the broader process:

## Key Goals in Generating Robust Embeddings:

### 1. Capture Intrinsic Graph Structure:

- The goal of generating robust embeddings is to ensure that the **embeddings** produced by the model effectively represent the **graph structure**—including both **node connectivity** (neighbors) and **edge types** (positive vs. negative).
- For signed graphs, capturing the **significance of positive and negative relationships** is crucial. A robust embedding will reflect not just the proximity of nodes in the graph, but

Message ChatGPT





- In **contrastive learning**, the aim is to learn representations that are consistent across different "views" of the graph. For each augmented view, the GNN takes the graph structure (with its modified edges or nodes) and learns an embedding.
- The GNN is key to ensuring that nodes from **positive pairs** (nodes that should be close in the embedding space due to positive relationships) are embedded similarly and **negative pairs** (,, s with a negative relationship) are embedded differently.
- The GNN processes these different views, producing embeddings that are invariant to the augmentations (i.e., robust across different transformations) but respect the signed relationships in the graph.

#### 4. Incorporating Attention Mechanisms:

- Many modern GNN architectures use **attention mechanisms** (e.g., Graph Attention Networks, GATs) to weigh the importance of neighboring nodes. This is particularly useful for signed graphs, as **positive edges** may be more influential than **negative edges**, and the GNN can learn to give different importance to neighbors connected via different types of edges (positive vs. negative).

Message ChatGPT





- The GNN processes these different views, producing embeddings that are invariant to the augmentations (i.e., robust across different transformations) but respect the signed relationships in the graph.

#### 4. Incorporating Attention Mechanisms:

- Many modern GNN architectures use **attention mechanisms** (e.g., Graph Attention Networks, GATs) to weigh the importance of neighboring nodes. This is particularly useful for signed graphs, as **positive edges** may be more influential than **negative edges**, and the GNN can learn to give different importance to neighbors connected via different types of edges (positive vs. negative).
- Attention helps the model focus on **relevant neighbors** and learn **fine-grained embeddings** by considering not just the presence of connections but also the nature (positive or negative) of those connections.

#### 5. Contrastive Objective Optimization:

- After obtaining embeddings for the graph nodes using a GNN, the contrastive learning framework takes over. The GNN-generated embeddings are used in a contrastive objective

Message ChatGPT



how we are making sure that while augmentation of graph we are not changing real relationships between the nodes

To ensure that the real relationships between nodes are not distorted during augmentation, we can take the following precautions:

1. **Edge Preservation in Augmentation:** When applying graph augmentations (such as node dropping, edge perturbation, or attribute masking), it's essential to carefully preserve the key relationships, especially the signed edges. For example, while dropping nodes or edges, we should avoid removing critical connections that might represent strong positive or negative relationships between nodes.
2. **Contextual Augmentation:** Augmentation techniques can be designed to respect the context of the signed relationships (positive/negative edges). For instance, if a positive edge indicates a strong relationship (friendship or cooperation), augmentations should avoid breaking such meaningful connections unless they serve to strengthen the robustness of the learned representation.
3. **Theory-driven Augmentation:** Signed graphs often rely on underlying theories (e.g., balance theory). Any augmentation should adhere to these theoretical constraints to ensure that the overall graph structure's semantics are preserved. For example, augmenting negative edges (enemies) may need to maintain the antisymmetric properties of signed edges.
4. **Contrastive Learning Consistency:** In contrastive learning, we aim to preserve the relationships through the contrastive loss by ensuring that positive pairs (connected nodes

Message ChatGPT





Using more than one GNN in the context of signed graph contrastive learning (SGCL) or similar tasks serves several purposes, particularly when the graph has complex, heterogeneous relationships like positive and negative edges. Here are some key reasons why multiple GNNs might be used:

### 1. Capturing Different Types of Relationships:

- In signed graphs, we typically have two types of relationships: **positive** and **negative** edges. Each of these relationships carries different semantics:
  - **Positive edges** represent friendly or cooperative relationships.
  - **Negative edges** represent antagonistic or adversarial relationships.
- By using multiple GNNs, we can dedicate one GNN to focus on positive relationships and another to focus on negative relationships, allowing the model to learn more nuanced and specialized embeddings for each type of connection.

### 2. Better Representation of Graph Structure:

- Different GNNs may focus on different aspects of the graph's structure, such as the local neighborhood or global graph structure. For example:
  - One GNN could learn embeddings based on the local neighborhood around a node.

Message ChatGPT



meaning of capturing invariant node representation

 **Capturing invariant node representation** refers to the idea of learning node embeddings (or features) that remain consistent and robust despite different transformations or augmentations of the graph. The key idea is that these representations should preserve the essential characteristics of the nodes and their relationships, regardless of changes that occur during the process of graph augmentation.

In the context of **contrastive learning** and **signed graphs**, this means:

### 1. Consistency Across Augmented Views:

- When we perform augmentations (like modifying edges or nodes in the graph), the generated **augmented graphs** should still represent the same underlying structure and relationships of the original graph. The goal is that the node embeddings should **capture these invariant features**—such as the role or importance of a node in the graph—without being overly sensitive to random changes made during augmentation.

### 2. Balance Theory in Signed Graphs:

- In **signed graphs**, there are positive and negative edges that represent different types of relationships (e.g., friendship vs. antagonism). The node representation should **preserve the sign of these relationships** (i.e., whether the connection is positive or negative) and

Message ChatGPT



## ChatGPT

Share

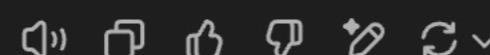
D

- In **signed graphs**, there are positive and negative edges that represent different types of relationships (e.g., friendship vs. antagonism). The node representation should **preserve the sign of these relationships** (i.e., whether the connection is positive or negative) and capture this balance across augmented views. For example, nodes connected by positive edges (friend relationships) should be close in the embedding space, while nodes connected by negative edges (enemy relationships) should be far apart.

### 3. Invariant Representations:

- The idea of **invariance** comes from the need to ensure that the **representations (embeddings)** of nodes are not affected by graph perturbations (like edge removal or addition). This guarantees that the embeddings capture the core structural properties of the node and its neighborhood in the graph, which are robust to variations in the graph's structure.

In short, **capturing invariant node representations** is about learning representations that are stable and preserve the essential structure and semantics of the graph, even when the graph undergoes transformations during the contrastive learning process.



Message ChatGPT



augmentations focus on perturbing balanced cycles, which are essential for maintaining the integrity of positive and negative links in the graph. Here's a breakdown:

## 1. Connectivity Perturbation

- **Description:** This augmentation involves perturbing the connectivity of the graph by randomly removing and adding links.
- **Process:**
  1. Randomly remove edges from the graph with a probability  $r$ .
  2. After removing edges, new edges are added to the graph:
    - $p$  positive edges are added.
    - $q$  negative edges are added.
- **Goal:** The aim is to test the robustness of the graph structure against changes in the connectivity while preserving the balance between positive and negative links. This approach assumes that the overall semantics of the graph are robust to variations in the structure of balanced cycles.

## 2. Sign Perturbation

- **Description:** Unlike connectivity perturbation, sign perturbation modifies the sign of the

Message ChatGPT



## 2. Sign Perturbation

- **Description:** Unlike connectivity perturbation, sign perturbation modifies the sign of the edges in the graph.
- **Process:**
  1. A positive edge is flipped to a negative edge with a probability  $r$ .
  2. A negative edge is flipped to a positive edge with the same probability.
- **Goal:** This approach assumes that small changes in the signs of the links (e.g., flipping positive links to negative and vice versa) will not significantly affect the model's predictions, but will still help in learning more robust and invariant node representations.

### Purpose of the Augmentations:

- Both augmentations focus on perturbing the existing balanced cycles. The key idea is to explore how changes to the graph (either in terms of connectivity or sign) affect the learning process, while keeping the number of links in the graph constant. These augmentations help in improving the generalization and robustness of the model by ensuring that it can still make accurate predictions even when the graph is slightly altered.

Message ChatGPT



what are balanced cycles here

“

In the context of signed graphs, **balanced cycles** refer to cycles (closed loops) in the graph where the signs of the edges follow a particular pattern, which is consistent with **balance theory**. Balance theory in social network analysis suggests that relationships in a signed graph are stable (or "balanced") if they follow a certain rule. Specifically:

- **Balanced Cycle:** A cycle is considered balanced if the product of the signs of the edges in the cycle is positive. This means that:
  - An even number of negative edges in the cycle (negative relationships) results in a balanced cycle.
  - An odd number of negative edges would make the cycle unbalanced.

### Example:

Consider a cycle with three nodes  $A$ ,  $B$ , and  $C$ , and the edges between them are signed (positive or negative):



- If  $A$  is connected to  $B$  by a positive edge (+),  $B$  is connected to  $C$  by a negative edge (-)

Message ChatGPT



does by perturbing we increase or decrease balanced cycles

“

By perturbing the graph, you can either increase or decrease the number of balanced cycles, depending on the type of perturbation applied and how the perturbations affect the relationships (edges) between nodes.

Here's how the two types of perturbations influence the balanced cycles:

## 1. Connectivity Perturbation:

- **Increase or Decrease Balanced Cycles:** In this method, random links are dropped and new links are added. The links that are added could either create new balanced cycles (by forming cycles where the product of edge signs is positive) or destroy existing balanced cycles (by altering the structure in such a way that cycles become unbalanced).
- For example, adding a link between two nodes that have opposite signs (positive and negative relationships) could potentially turn a balanced cycle into an unbalanced one, or vice versa, depending on the rest of the graph structure.

## 2. Sign Perturbation:



- **Increase or Decrease Balanced Cycles:** By changing the signs of existing edges, you could

Message ChatGPT



number of negative edges) into an unbalanced one (odd number of negative edges), or vice versa.

### Summary:

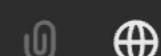
,,

- **Goal of Perturbation:** The main goal of perturbation in the context of signed graphs is to explore and expose different patterns of relationships that the model might not have seen before. This could lead to the identification of more balanced cycles (increasing robustness) or the elimination of unimportant or spurious ones (leading to cleaner, more generalizable structures).
- **Effect:** Perturbations aim to *enhance the model's robustness and generalization*. While the perturbations might decrease the number of balanced cycles in some cases, they could also discover new, meaningful cycles that were not present in the original graph structure, improving the overall quality of learned embeddings.

So, **perturbation can both increase or decrease the number of balanced cycles**, but the focus is on improving the model's ability to generalize by exploring different graph structures.



Message ChatGPT



representations that capture the relationships between "enemies" (negative connections).

#### 4. Representation Types:

- **Consistent Representations:** The node representations learned from the positive graphs are called **consistent representations**, as they reflect the relationships that are positively related (friendship).
- **Inconsistent Representations:** The node representations learned from the negative graphs are called **inconsistent representations**, as they reflect the relationships that are negatively related (enmity).

### Why Two Separate GNNs?

The reason for using two separate GNNs is rooted in the fact that positive and negative links carry **distinct semantics**:

- **Positive links** represent **friendly relationships** (nodes are similar or related).
- **Negative links** represent **hostile relationships** (nodes are dissimilar or opposed).

By separately processing these two types of 's with different GNNs, the model can learn **more expressive and nuanced node embeddings** that respect these different types of relationships.

Message ChatGPT



negative sign).

### 3. Positive and Negative Graph Neural Networks:

- The idea behind using two separate GNNs is to separately handle the positive and negative relationships:
  - Positive GNN:** A GNN is applied to the positive graph ( $G_i^+$ ) to learn node representations that capture the relationships between "friends" (positive connections).
  - Negative GNN:** Another GNN is applied to the negative graph ( $G_i^-$ ) to learn node representations that capture the relationships between "enemies" (negative connections).

### 4. Representation Types:

- Consistent Representations:** The node representations learned from the positive graphs are called **consistent representations**, as they reflect the relationships that are positively related (friendship).
- Inconsistent Representations:** The node representations learned from the negative graphs are called **inconsistent representations**, as they reflect the relationships that are negatively related (enmity).



Message ChatGPT



## Graph Encoder Process:

### 1. Graph Augmentations:

- **Two Graph Views ( $G_1, G_2$ ):** After performing graph augmentations (e.g., connectivity and sign perturbation), two distinct views of the graph are generated, each representing a slightly altered version of the original signed graph.

### 2. Splitting Graph Views:

- For each graph view  $G_i$ , the augmented graph is split into two graphs:
  - **Positive Graph ( $G_i^+$ ):** This graph contains only the positive links (edges with a positive sign).
  - **Negative Graph ( $G_i^-$ ):** This graph contains only the negative links (edges with a negative sign).

### 3. Positive and Negative Graph Neural Networks:

- The idea behind using two separate GNNs is to separately handle the positive and negative relationships:
  - **Positive GNN:** A GNN is applied to the positive graph ( $G_i^+$ ) to learn node representations that capture the relationships between "friends" (positive connections).

Message ChatGPT



## Key Points:

### 1. Aggregation Process:

- The **aggregation layer** is designed to update a node's representation by aggregating information from its neighbors.
- Since the contribution of neighbors to the node's representation may vary, the model uses an **attention mechanism** to assign different importance to each neighbor.

### 2. Attention Mechanism:

- For each node  $v_i$ , its **neighbors**  $v_j$  in the graph (either positive or negative) will contribute to its updated representation.
- The attention mechanism assigns an **attention coefficient**  $\alpha_{ij}$  to each neighbor  $v_j$ , which is calculated using a weight vector  $a^+$  (for the positive graph) or  $a^-$  (for the negative graph).

### 3. Attention Coefficient Calculation (Equation 1 & 2):

- For each node  $v_i$  in the graph, the attention coefficient  $\alpha_{ij}$  for its neighbor  $v_j$  is computed using the following steps:
  - Compute a **LeakyReLU** activation of the concatenation of the node's representation  $h_i$  and its neighbor's representation  $h_j$ . (both representations are multiplied by their

Message ChatGPT



- The attention mechanism assigns an **attention coefficient**  $\alpha_{ij}$  to each neighbor  $v_j$ , which is calculated using a weight vector  $a^+$  (for the positive graph) or  $a^-$  (for the negative graph).

### 3. Attention Coefficient Calculation (Equation 1 & 2):

- For each node  $v_i$  in the graph, the attention coefficient  $\alpha_{ij}$  for its neighbor  $v_j$  is computed using the following steps:
  - Compute a **LeakyReLU** activation of the concatenation of the node's representation  $h_i$  and its neighbor's representation  $h_j$  (both representations are multiplied by their respective weight matrices  $W_a^+$  or  $W_a^-$ ).
  - The **attention coefficient**  $\alpha_{ij}$  is then computed as:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(a^+[h_i W_a^+ \| h_j W_a^+]^T))}{\sum_{t \in N_i^+} \exp(\text{LeakyReLU}(a^+[h_i W_a^+ \| h_t W_a^+]^T))}$$

for the **positive graph** and similarly for the **negative graph** (Equation 2).

- Here:
  - $N_i^+$  and  $N_i^-$  are the sets of neighbors of node  $v_i$  in the positive and negative graphs, respectively.



Message ChatGPT



- $W_a^+$  and  $W_a^-$  are the weight matrices used for the positive and negative graphs.
- The LeakyReLU nonlinearity is applied to ensure the model can learn complex relationships and avoid issues with vanishing gradients. It has a negative slope  $\alpha = 0.2$ .
- The attention coefficient  $\alpha_{ij}$  is normalized across all neighbors to ensure they sum to 1 (for each node).

#### 4. Concatenation and Weighting:

- The input node's representation and the neighbor's representation are **concatenated** (denoted by  $||$ ) before applying the attention mechanism.
- The attention mechanism assigns higher weights (importance) to neighbors that are more relevant (i.e., more strongly connected), and these coefficients help to update the node's final representation.

#### Summary:

- The attention mechanism helps decide how much influence each neighbor should have in updating the node's representation.
- For each node and its neighbors, the attention coefficient is calculated by combining the representations of the node and the neighbor, followed by a LeakyReLU activation.

Message ChatGPT

