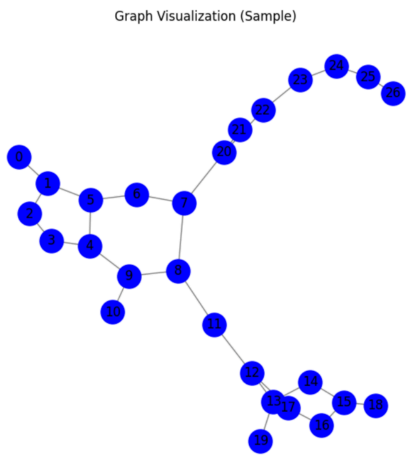
Part3-graph-neural-networks-concept

Project: Explainable AI-Driven Adverse Drug Reactions Prediction Toward Pediatric Drug Discovery & Development

Graph-Based and Molecular Representations:

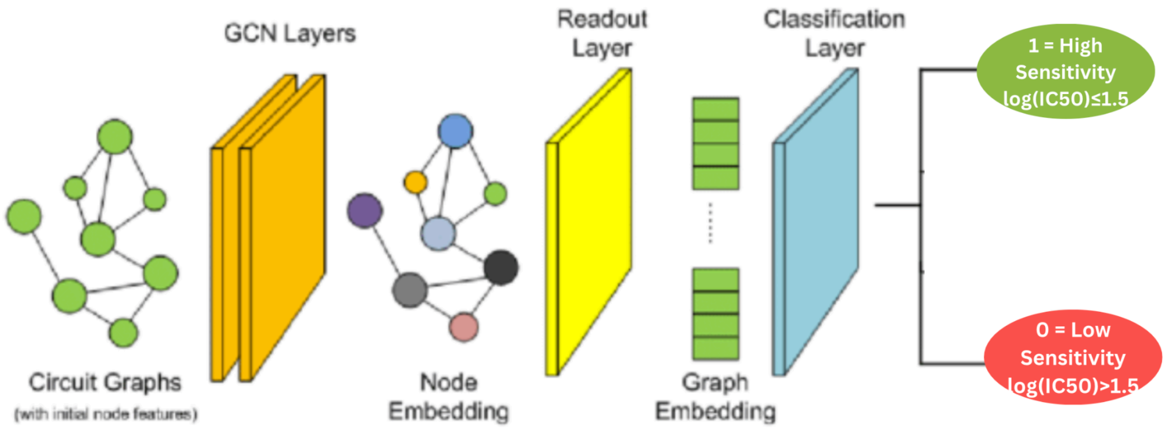
Graph Neural Networks (GNNs) and knowledge graphs are powerful tools for modeling the complex, non-Euclidean (**doesn't lie on a regular grid** like images or tables) structure of biological systems. In contexts such as gene-drug or protein-pathway interactions, biological entities (e.g., genes, proteins, compounds, diseases) can be represented as **nodes**, while relationships such as binding, regulation, inhibition, or co-expression are encoded as **edges**.

Molecular graphs for representing compounds



```
Atoms: ['C', 'N', 'C', 'N', 'C', 'O', 'N', 'C', 'C', 'C', 'C', 'O', 'C', 'O', 'O', 'O', 'N']
Bonds: [(0, 1, rdkit.Chem.rdchem.BondType.AROMATIC), (1, 2,
rdkit.Chem.rdchem.BondType.AROMATIC), (2, 3,
rdkit.Chem.rdchem.BondType.AROMATIC), (3, 4,
rdkit.Chem.rdchem.BondType.AROMATIC), (4, 5,
rdkit.Chem.rdchem.BondType.DOUBLE), (4, 6,
rdkit.Chem.rdchem.BondType.AROMATIC), (6, 7,
rdkit.Chem.rdchem.BondType.SINGLE), (7, 8,
rdkit.Chem.rdchem.BondType.SINGLE), (8, 9,
rdkit.Chem.rdchem.BondType.SINGLE), (9, 10,
rdkit.Chem.rdchem.BondType.SINGLE), (10, 11,
rdkit.Chem.rdchem.BondType.SINGLE), (10, 12,
rdkit.Chem.rdchem.BondType.SINGLE), (12, 13,
rdkit.Chem.rdchem.BondType.SINGLE), (9, 14,
rdkit.Chem.rdchem.BondType.SINGLE), (8, 15,
rdkit.Chem.rdchem.BondType.SINGLE), (2, 16,
rdkit.Chem.rdchem.BondType.SINGLE), (6, 0,
rdkit.Chem.rdchem.BondType.AROMATIC), (11, 7,
rdkit.Chem.rdchem.BondType.SINGLE)]
```

For example, the previous image provides a concrete example of a **molecular graph** constructed from a chemical compound. Here, each node represents an atom (e.g., 'C', 'N', 'O'), and each edge corresponds to a chemical bond (e.g., SINGLE, DOUBLE, AROMATIC), as generated using RDKit. This type of molecular graph **can serve as the input to the GNN** in the image below, allowing the model to learn structure-activity relationships by encoding both atom types and bond connectivity. Such representations are especially useful in drug discovery tasks where understanding the effect of molecular substructures on biological activity is critical.


GCN Layers
Readout Layer
Classification Layer
1 = High Sensitivity log(IC50)≤1.5
0 = Low Sensitivity log(IC50)>1.5
Circuit Graphs (with initial node features)
Node Embedding
Graph Embedding

(https://www.researchgate.net/figure/Architecture-of-GNN-model-for-circuit-classification_fig1_356624687)

Once scientists build these connection maps showing how genes, proteins, and drugs relate to each other, they put them through a smart computer system. This system works like passing notes in a classroom - each point on the map (like a gene) starts knowing only about itself, but then shares information with its neighbors. After several rounds of this information sharing, each point understands not just itself but also what's happening around it in the bigger picture.

For example, when trying to figure out if a drug will work against cancer, the computer doesn't just look at one gene alone. It considers that gene plus all the genes connected to it and any drugs that target those genes. This gives a much more complete view of how everything works together.

After all this information sharing, the system combines everything it learned into one big summary that captures both the individual pieces and how they connect. Finally, it makes a prediction - will this drug work (yes) or not work (no) for this patient's specific situation.

To make this even smarter, scientists also feed in knowledge from medical databases - like giant libraries of information about how genes cause diseases, what drugs target which proteins, and how biological processes work. This is like giving the computer access to textbooks while it's solving the problem.

By combining what it learns from the data with what's already known from years of research, the computer can make better predictions about new drug treatments, discover which genes might cause diseases, or figure out the most important control points in how cells work. This approach is especially valuable for drug discovery because the predictions are based on real biological knowledge, making them more trustworthy and useful for developing new medicines.[more detailed links].

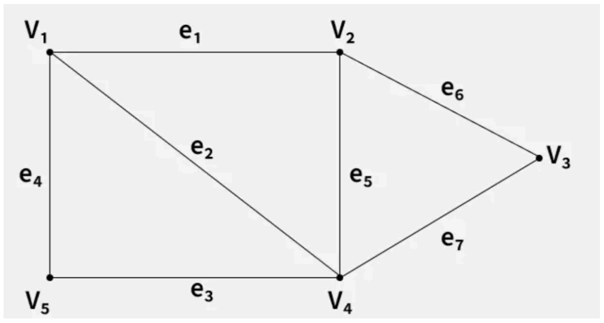
Explain the mathematical foundations of graph reasoning in the context of biological networks. Compare key concepts such as k-hop neighborhoods, meta-paths, subgraphs, and attention mechanisms. How do these approaches differ in their ability to capture local versus global relationships among biological entities?

Graph Neural Networks (GNNs) effectively model biological systems by representing entities like atoms, genes, and proteins as nodes, with edges capturing their relationships. These models predict molecular properties and drug responses by leveraging both individual features and network structure through techniques like k-hop neighborhoods, meta-paths, subgraphs, and attention mechanisms:

Sub- graph:

Biological systems such as gene regulatory networks, protein–protein interaction networks, and metabolic pathways can be modeled as **graphs** G=(V,E) where:

- V: nodes (e.g., genes, proteins, compounds, diseases)
- E: edges (e.g., interactions, regulations, bindings)

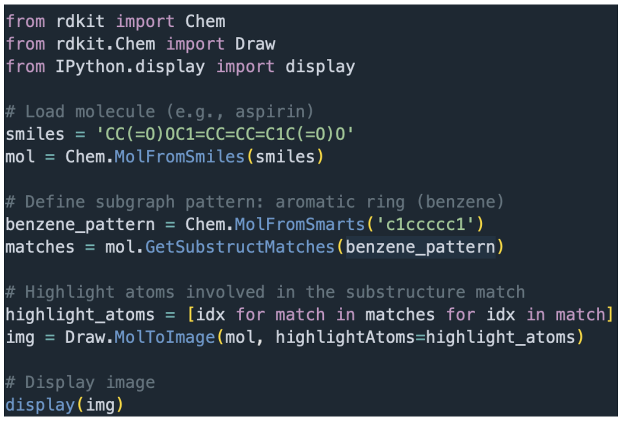


(https://www.geeksforgeeks.org/graph-types-and-applications/)

A **subgraph** is simply a **smaller part of a larger graph**.
If graph **G1 = (V1, E1)** is a subgraph of **G = (V, E)**, then:

- **V1** ⊆ **V** (nodes in the subgraph are part of the main graph)
- **E1** ⊆ **E** (edges in the subgraph also exist in the main graph)
- The edges **must connect the same nodes** as they do in the main graph.

Example: In the aspirin molecule, the **benzene ring** is a subgraph, a functional group within the full chemical graph of aspirin.



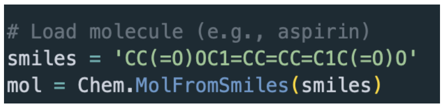
```python
from rdkit import Chem
from rdkit.Chem import Draw
from IPython.display import display

# Load molecule (e.g., aspirin)
smiles = 'CC(=O)OC1=CC=CC=C1C(=O)O'
mol = Chem.MolFromSmiles(smiles)

# Define subgraph pattern: aromatic ring (benzene)
benzene_pattern = Chem.MolFromSmarts('c1ccccc1')
matches = mol.GetSubstructMatches(benzene_pattern)

# Highlight atoms involved in the substructure match
highlight_atoms = [idx for match in matches for idx in match]
img = Draw.MolToImage(mol, highlightAtoms=highlight_atoms)

# Display image
display(img)
```

https://www.rdkit.org/docs/RDKit_Book.html

Graph Representation of Molecules (Using RDKit), In cheminformatics, molecules like aspirin can be represented as graphs:

- **Atoms** → nodes (V)
- **Bonds** → edges (E)

For example:



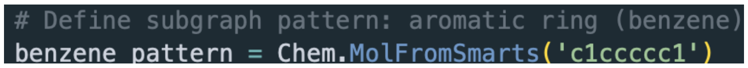
```python
# Load molecule (e.g., aspirin)
smiles = 'CC(=O)OC1=CC=CC=C1C(=O)O'
mol = Chem.MolFromSmiles(smiles)
```

- Aspirin has the SMILES code "CC(=O)Oc1ccccc1C(=O)O"
- RDKit can convert this into a molecular graph

Subgraph Pattern Matching:

To find a specific part (like a benzene ring) inside a molecule:

1. Define the pattern you want to find — this is the query graph Q = (V′, E′)
   - For a benzene ring, this is typically described using SMARTS notation: "c1ccccc1"



```python
# Define subgraph pattern: aromatic ring (benzene)
benzene_pattern = Chem.MolFromSmarts('c1ccccc1')
```

Subgraph Matching Goal:

1. Find a mapping **f: V′ → V** from the query graph to the molecule such that:
   a. Each atom in the query matches an atom in the molecule (same type, same bonds)
   b. Each bond in the query matches a bond in the molecule

   c. This calls RDKit's built-in subgraph isomorphism search (e.g., VF2 algorithm). It returns a list of valid mappings:

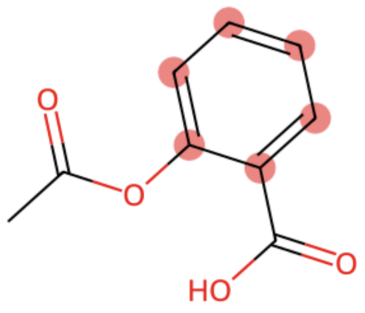
```
matches = mol.GetSubstructMatches(benzene_pattern)
```

   d. Output : matches=[(f(v1),f(v2),…,f(v6))]

2. Each tuple is a valid matching of the benzene ring in the molecule, next extract all atom indices involved:

```
# Highlight atoms involved in the substructure match
highlight_atoms = [idx for match in matches for idx in match]
```

3. After finding the matching atoms and bonds:
    - RDKit can **highlight** the subgraph directly on the molecular image.

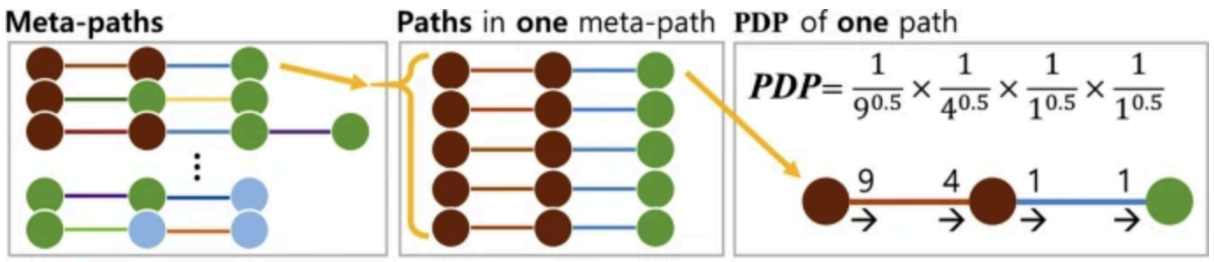


   - In the aspirin example, the **benzene ring is shown with red circles**, marking a successful match.

This formulation provides a strict mathematical basis for **pattern matching by ensuring that only patterns with identical labels and connectivity are considered a match.** Such precision is invaluable in drug discovery for pinpointing functional groups that bind to proteins, in molecular fingerprinting for identifying similar compounds, and in explainable AI for revealing the substructures that influence a model's predictions.

Meta-paths in Graph

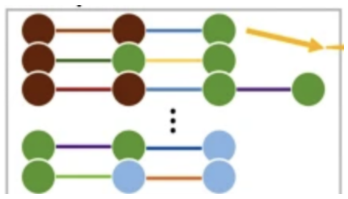
A **meta-path** is a sequence of node types and edge types in a **heterogeneous graph**. It formalizes **semantic relations** between entities.


Meta-paths
Paths in one meta-path
PDP of one path
$PDP = \frac{1}{9^{0.5}} \times \frac{1}{4^{0.5}} \times \frac{1}{1^{0.5}} \times \frac{1}{1^{0.5}}$
9
4
1
1

(https://www.nature.com/articles/s41598-019-41814-w)

1. Meta-paths = Biological Relationship Templates (Left Panel)

A **meta-path** defines the **sequence of biological entity types** that want to explore.



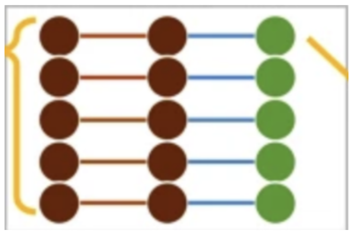
Common biological meta-paths, For instance:

- **Drug → Protein → Gene → Disease** (How does a drug affect disease through molecular targets?)
- **Gene → Pathway → Phenotype** (How do genes influence observable traits?)
- **Protein → Protein → Function** (How do protein interactions determine cellular functions?)

Each colored node represents a different **type** of biological entity (drug, gene, protein, disease, etc.), and the template shows the **sequence of biological relationships** we want to investigate.[links]

2. Actual Paths = Real Biological Connections (Middle Panel)

From **one meta-path template**, we can find **many real biological pathways** in the dataset.



From "Drug → Protein → Gene → Disease" , For instance:

- Drug_A → ProteinX → GeneAlpha → DiseaseType1
- Drug_B → ProteinY → GeneBeta → DiseaseType2
- Drug_C → ProteinZ → GeneGamma → DiseaseType3
- Drug_D → ProteinW → GeneDelta → DiseaseType4

Each row represents a **specific biological pathway** following the same structural pattern, where we can analyze how different drugs interact with their target proteins, which are encoded by specific genes, ultimately affecting various disease outcomes.

3. PDP = Biological Specificity Score (Right Panel)

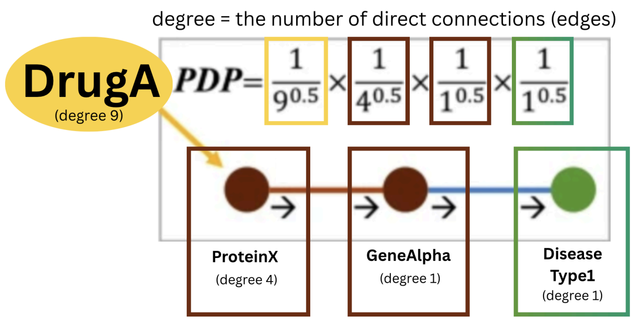
PDP (Path Degree Product) measures how **specific** and informative a pathway is in the biological network.

The Scoring System:

$$PDP = \frac{1}{9^{0.5}} \times \frac{1}{4^{0.5}} \times \frac{1}{1^{0.5}} \times \frac{1}{1^{0.5}}$$

- Each fraction represents **one node in the pathway**
- The number in the denominator is that node's **degree** (number of connections)
- The **0.5 exponent** provides a square root penalty
- **Multiply all terms** to get the final pathway score

In this pathway example:


degree = the number of direct connections (edges)
DrugA
(degree 9)
$PDP = \frac{1}{9^{0.5}} \times \frac{1}{4^{0.5}} \times \frac{1}{1^{0.5}} \times \frac{1}{1^{0.5}}$
ProteinX
(degree 4)
GeneAlpha
(degree 1)
Disease Type1
(degree 1)

- **DrugA** (degree 9): Very common drug with many targets → contributes 1/3 ≈ 0.33 to the product
- **ProteinX** (degree 4): Moderately connected protein → contributes 1/2 = 0.5 to the product
- **GeneAlpha** (degree 1): Specific gene → contributes 1/1 = 1.0 to the product
- **DiseaseType1** (degree 1): Specific disease → contributes 1/1 = 1.0 to the product
- **Final PDP** = 0.33 × 0.5 × 1.0 × 1.0 ≈ 0.17

Mathematical interpretation:

- **Higher degree** = smaller fraction = lower PDP score
- **Lower degree** = larger fraction = higher PDP score
- The **product** ensures that even one highly connected hub dramatically reduces the pathway's importance

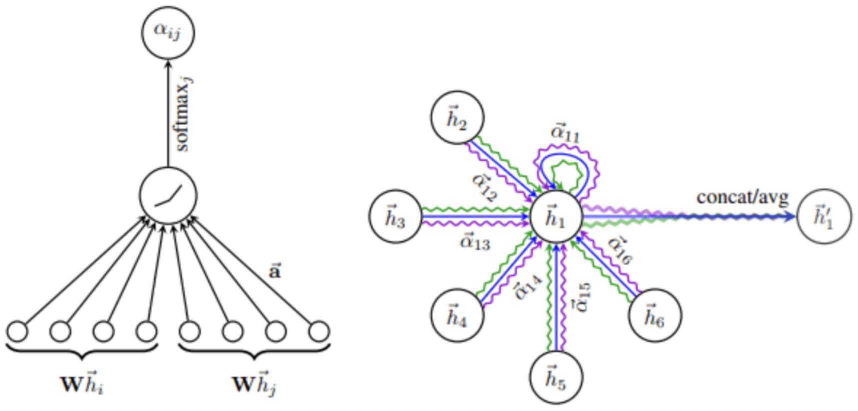
Biological interpretation:

- **Highly connected nodes (hubs) get penalized -** like common proteins that interact with many entities
- **Specific, rare connections get rewarded -** like specialized proteins or rare disease-specific genes

This system helps researchers identify **specific, clinically relevant biological pathways** for drug discovery and disease understanding, rather than focusing on obvious but non-specific molecular interactions.

Graph Attention Networks (GATs) in Molecular Representation

Graph Attention Networks (GATs) extend traditional Graph Neural Networks by allowing nodes to learn the **relative importance of their neighbors** using attention mechanisms. In biological networks, such as protein-protein interaction graphs or gene regulatory networks, not all neighboring biological entities contribute equally to a prediction task. GATs enable the model to dynamically focus on the most informative molecular interactions.



Attention in GAT. Left: The attention mechanism. Right: An illustration of multihead attention on its neighborhood. Source: Graph Attention Networks[6]

https://theaisummer.com/gnn-architectures/

Implementation with example

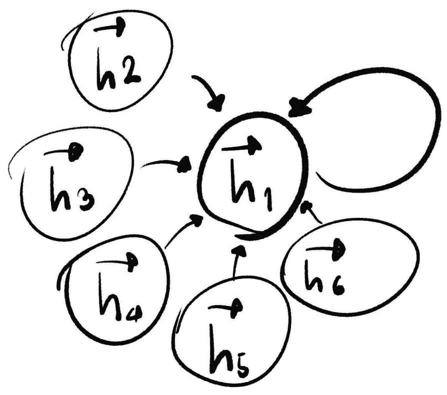
1: Prepare Input Node Features

Each biological entity (e.g., gene, protein, or compound) is represented as a node with an associated feature vector. These features might include: - Atom types or chemical properties (in molecules) - Gene expression levels (in transcriptomic data) - Structural or sequence features (for proteins)

- Each node vi has an initial feature vector:

$$\vec{h}_i \in \mathbb{R}^F$$

where F is the number of input features.R*F* denotes a vector of F real-valued features.



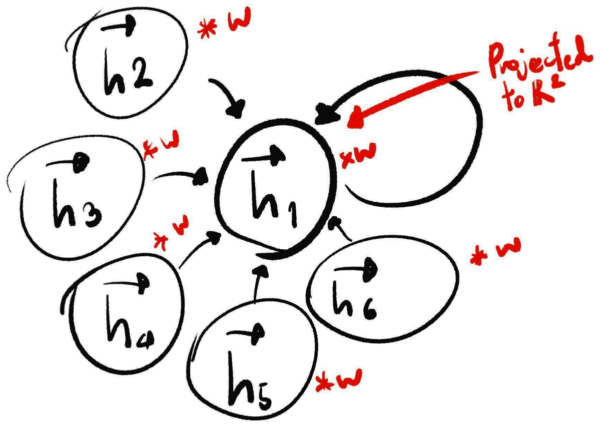
- These features could be chemical properties (in molecules), gene expressions (in biology), etc.

2.Linear Transformation

To prepare for attention computation, each node's features are linearly transformed into a new feature space:

$$\vec{h}' = W\vec{h}_i$$

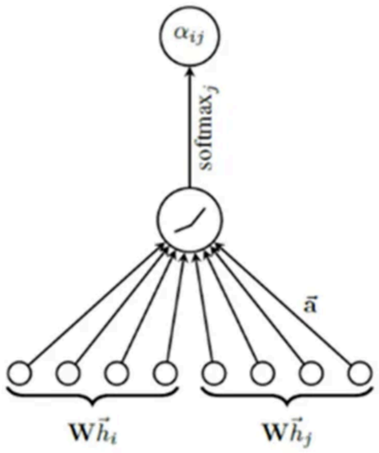
thus:



This allows the model to learn a better representation of each biological node in a unified space.

3: Compute Attention Coefficients

To capture the importance of each neighboring node, a shared attention mechanism computes a raw score *e_ij* for each edge (*i*, *j*) :



3.1. Concatenate Transformed Features: For each node *i* and its neighbor *j* ∈ N (*i*) ∪ {*i*} , concatenate their transformed feature vectors:

$$[Wh_i||Wh_j]$$

3.2. Scoring Function: Apply a shared attention vector *aT* followed by a LeakyReLU nonlinearity:

$$e(h_i, h_j) = LeakyReLU(a^T[Wh_i||Wh_j])$$

https://medium.com/stanford-cs224w/fraud-detection-with-gat-edac49bda1a0

- **Where:**
  - **||** = vector concatenation
  - **T** = transpose
  - **a** = a learnable attention vector, a row vector (1D array) of weights.
  - **a**$^{\mathbf{T}}$ = the transpose of that vector, turning it into a column vector.

This computes a raw attention score *eij* indicating the importance of node *j* 's features to node *i* .

For each edge (i, j), compute:

- e_ij = LeakyReLU(a^T [W*h_i || W*h_j])

Thus

- e_{11} = LeakyReLU(a^T [W h_1 || W h_1]) ← self-loop
- e_{12} = LeakyReLU(a^T [W h_1 || W h_2]) ← from h_2
- e_{13} = LeakyReLU(a^T [W h_1 || W h_3]) ← from h_3
- e_{14} = LeakyReLU(a^T [W h_1 || W h_4]) ← from h_4
- e_{15} = LeakyReLU(a^T [W h_1 || W h_5]) ← from h_5
- e_{16} = LeakyReLU(a^T [W h_1 || W h_6]) ← from h_6

4: Normalize Attention with Softmax

Once you have all e_ij for every neighbor j of node i, you normalize them using **softmax** across the neighborhood:

$$\alpha_{ij} = softmax(e(h_i, h_j) = \frac{exp(e(h_i, h_j)}{\sum_{j' \epsilon N_i}(exp(e(h_i, h_j')}$$

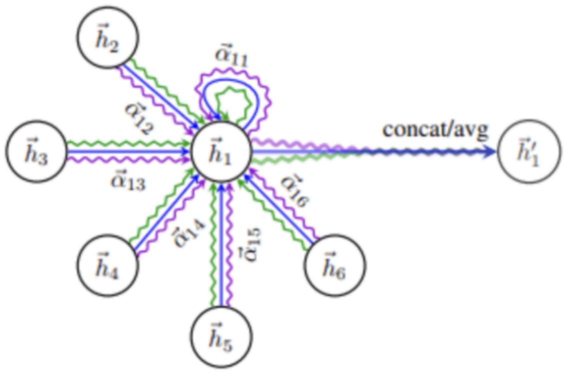
- This produces attention coefficients αij∈(0,1)
- They now **sum to 1** across all neighbors (i.e., ∑jα_ij=1)
- They define how much influence each neighbor's information will have on node i
    - where for center node h1:
        - α_{11} = softmax(e_{11}) = exp(e_{11}) / Σ_k exp(e_{1k}) ← self-loop
        - α_{12} = softmax(e_{12}) = exp(e_{12}) / Σ_k exp(e_{1k}) ← from h_2
        - α_{13} = softmax(e_{13}) = exp(e_{13}) / Σ_k exp(e_{1k}) ← from h_3
        - α_{14} = softmax(e_{14}) = exp(e_{14}) / Σ_k exp(e_{1k}) ← from h_4
        - α_{15} = softmax(e_{15}) = exp(e_{15}) / Σ_k exp(e_{1k}) ← from h_5
        - α_{16} = softmax(e_{16}) = exp(e_{16}) / Σ_k exp(e_{1k}) ← from h_6

5: Aggregate Neighbor Information

After computing attention weights αij, next using them to combine the neighbors' feature vectors and update the center node hi's representation.

$$h_i^{(l)} = \sigma(\sum_{j \epsilon N(i)} \alpha_{ij} W^{(l)} h_j^{(l-1)})$$

- Where:
  - αij: attention weight from neighbor $j \rightarrow i$
  - $W\vec{h}j$: transformed feature vector of neighbor jj
  - $\sum$: sum over all neighbors of ii, including itself
  - σ: activation function (e.g., ELU or ReLU)
- Thus



$$h_1' = \sigma((\alpha_{11}\times Wh_1)+(\alpha_{12}\times Wh_2) + (\alpha_{13}\times Wh_3)+(\alpha_1\times Wh_4)+(\alpha_{15}\times Wh_5) + (\alpha_{16}\times Wh_6))$$

What Happens Next:

- Once Gene 1 has this updated, smarter representation, scientists can use it for different jobs:
- Identifying genes: "Is this gene involved in cancer?" (like sorting genes into cancer-related vs. normal)
- Testing drugs: "Will this drug be toxic?" (by looking at the whole drug network)
- Finding connections: "Does this drug target this protein?" (predicting new relationships)

If you stack multiple layers of this process, it's like playing telephone - each round, the genes learn about neighbors that are further and further away, eventually understanding whole biological pathways instead of just their immediate neighborhood.

Why This Is Special: Unlike older computer models that treated all connections as equally important, this system is smart enough to pay more attention to the connections that actually matter. In biology, some gene interactions are much more important than others for things like drug response. This attention system automatically figures out which relationships are most important and focuses on those, kind of like a detective who knows which clues are worth investigating. For drug discovery, this means the computer can point to exactly which parts of the cell are most important for a drug's effect, making the predictions both more accurate and easier for scientists to understand.

K-hop neighborhoods:

Each layer in GAT works like expanding circles of influence. In the first layer, a gene only "talks to" its direct partners (1-hop neighbors). In the second layer, it learns about genes that are 2 steps away through its original partners. By the third layer, it knows about genes 3 steps away, and so on. This is called a k-hop neighborhood, where k represents how many steps away you're willing to look.

1: Gather Information (AGGREGATE): This is like conducting a survey of your biological neighborhood. The system:

$$a_v^{(t)} = \mathrm{AGGREGATE}^{(t)}\left(\left\{h_u^{(t-1)} | u \in \mathcal{N}_1(v)\right\}\right)$$

(https://www.researchgate.net/publication/334478270_k-hop_Graph_Neural_Networks)

- This equation describes how a node v at time step t **aggregates** information from its neighbors u∈Nk(v) the set of all nodes within k-hops.
- hu(t−1) is the **previous round's state** (e.g., gene expression, pathway involvement, etc.)
- The AGGREGATE function could be mean, sum, attention-weighted average, etc.

2: Update Knowledge (MERGE): This is where the magic happens - combining outside information with internal knowledge:

$$h_v^{(t)} = \mathrm{MERGE}^{(t)}\left(h_v^{(t-1)}, a_v^{(t)}\right)$$

(https://www.researchgate.net/publication/334478270_k-hop_Graph_Neural_Networks)

- This combines the node's **own past knowledge** hv(t−1) with the **aggregated neighborhood information** av(t).
- The MERGE step helps retain self-identity while incorporating context.

How k-hop Learning Works:

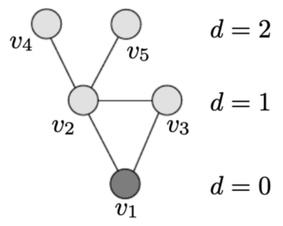
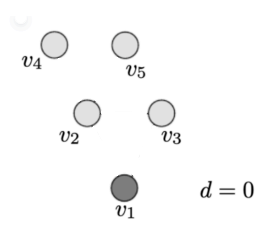


Figure 3: The 2-hop neighborhood graph $G^2_{v_1}$ of a node v_1 of graph G.

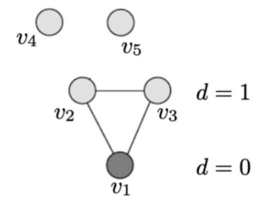
(https://www.researchgate.net/publication/334478270_k-hop_Graph_Neural_Networks)

k=0 (d=0): Gene v_1 alone



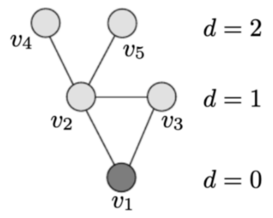
- Only considers *BRCA1*'s (short for *Breast Cancer gene 1*) own features (expression level, mutation status)
- Limited view: "I'm a DNA repair gene with high expression"

k=1 (d=1): 1-hop neighborhood {v$_2$, v$_3$}



- *BRCA1* learns from direct partners *TP53* and *ATM*
- Richer view: "I'm a DNA repair gene, and my direct partners are also involved in DNA damage response"
- **AGGREGATE**: Combines information from *TP53* (tumor suppression) + *ATM* (DNA damage sensing)
- **MERGE**: Updates *BRCA1*'s representation with this direct regulatory context

k=2(d=2): 2-hop neighborhood {v$_2$, v$_3$, v$_4$, v$_5$}



- *BRCA1* now learns from the entire network shown
- Complete view: "I'm part of a larger DNA repair pathway network"
- **AGGREGATE**: Now includes *CHEK2* (cell cycle checkpoint) + *RAD51* (homologous recombination)
- **MERGE**: *BRCA1* gets updated representation that captures its role in the broader DNA repair ecosystem

Biological Insight:

- **k=1**: Captures direct molecular interactions (protein-protein binding, direct transcriptional regulation)
- **k=2**: Captures pathway-level relationships (how DNA damage signals propagate through multiple genes)

This balance allows GNNs to make predictions that consider both the specific molecular details and the broader biological system, making them powerful for tasks like predicting which drugs will work for specific patients based on their complete biological profile.

Summary table:

Concept	Type of Information Captured Scope (Local/Global)	Scope (Local/Global)	Example case in Biology
K-hop	Structural relationships via graph distance	Local to semi-global (based on k)	Exploring nearby biological influence (e.g., how a gene's neighbors regulate its function)
Meta-path	Semantic relationships across node types	Global	Reasoning over heterogeneous biological pathways (e.g., drug → protein → gene → disease)
Subgraph	Structural motifs and patterns	Local	Detecting functionally meaningful structures (e.g., aromatic rings in drug molecules)
Attention(GAT)	Learned importance of neighbors	Both (adaptive per node)	Identifying key influencers in networks (e.g., prioritizing gene–gene interactions in prediction)

Where:

- **Local:** Refers to directly connected nodes (1-hop) or small neighborhoods;
- **Semi-global:** Includes nodes up to 2–3 hops away (moderately wide context)
- **Global:** Captures long-range, often cross-entity connections that span multiple steps and node types in the graph
- **Both:** GATs adaptively learn whether to prioritize local or more distant nodes by assigning learned attention weights
- **Semantic:** Refers to the types and meanings of nodes/edges involved (e.g., Drug vs. Gene vs. Disease), not just connections
- **Structural:** Refers to the shape or topology of the graph — who connects to whom, and how

https://towardsdatascience.com/a-unified-view-of-graph-neural-networks-12b40e8fdac5/

https://arxiv.org/pdf/1901.00596