# Chương 1. Tổng quan về mạng nơ-ron đồ thị và ứng dụng

## 1. Khái quát về mạng nơ-ron đồ thị

Mạng nơ-ron đồ thị (Graph Neural Network – GNN) là một lớp mô hình học sâu (deep learning) đặc biệt được thiết kế để xử lý dữ liệu có cấu trúc đồ thị, trong đó các đỉnh (nodes) biểu diễn các thực thể và các cạnh (edge) biểu diễn mối quan hệ giữa chúng. Khác với mạng nơ-ron truyền thống chỉ xử lý dữ liệu dạng lưới (như ảnh) hoặc chuỗi (như văn bản), GNN khai thác cấu trúc quan hệ phức tạp giữa các đỉnh thông qua cơ chế lan truyền thông tin (message passing). Ở mỗi lớp của GNN, mỗi đỉnh sẽ thu thập thuộc tính (feature) từ chính nó và các đỉnh lân cận, sau đó thực hiện phép kết hợp (aggregation) theo các hàm như tổng, trung bình hoặc hàm chú ý (attention) để tạo ra biểu diễn mới cho đỉnh đó. Quá trình này lặp lại qua nhiều lớp, giúp mỗi đỉnh tích hợp thông tin từ một vùng lân cận ngày càng rộng hơn, từ đó nâng cao khả năng học hiểu cấu trúc và tính chất toàn cục của đồ thị.

A collage of different types of information

Description automatically generated

**Hình 1.1:** Các miền ứng dụng của GNN

Có nhiều biến thể của GNN được đề xuất nhằm tối ưu hóa hiệu quả tính toán và khả năng biểu diễn, trong đó nổi bật nhất là Graph Convolutional Network (GCN), sử dụng phép tích chập mở rộng lên đồ thị để thu thập thông tin lan truyền; Graph Attention Network (GAT), kết hợp cơ chế chú ý để gán trọng số khác nhau cho các đỉnh lân cận; GraphSAGE, học các hàm ghép (aggregator) để tổng quát hóa tốt hơn cho các đồ thị chưa xác định... Các mô hình này ngày càng được cải tiến với các thành phần như skip connections, normalization, và các kỹ thuật regularization để giải quyết hiện tượng “over-smoothing” khi số lớp quá sâu. Bên cạnh đó, việc mở rộng GNN sang các đồ thị động, đồ thị con (subgraph), đồ thị với siêu đỉnh (hypergraphs) và đồ thị hỗn hợp (heterogeneous graphs) giúp mô hình thích ứng với nhiều dạng quan hệ phức tạp và dữ liệu thực tế hơn.

Trong quá trình huấn luyện, GNN thường áp dụng phương pháp học có giám sát (supervised learning) với hàm mất mát dựa trên nhiệm vụ (task-specific loss), ví dụ như phân loại đỉnh, phân loại cạnh hoặc dự đoán nhãn đồ thị. Ngoài ra, học bán giám sát (semi-supervised) cũng là xu hướng phổ biến, tận dụng cả các đỉnh đã gán nhãn và chưa gán nhãn thông qua kỹ thuật propagation label. Đối với các ứng dụng đòi hỏi suy luận cấu trúc, GNN có thể kết hợp với mô hình sinh (generative models) như VAE hoặc GAN để mô phỏng và tạo ra đồ thị mới.

GNN đã chứng minh hiệu quả vượt trội trong nhiều lĩnh vực như phân tích mạng xã hội (dự đoán liên kết, phát hiện cộng đồng…), sinh học (dự đoán tương tác protein-protein, xác định cấu trúc phân tử…), hóa học tính toán (dự đoán tính chất vật liệu, thuốc…), hệ thống khuyến nghị (recommendation systems), và thậm chí trong xử lý ngôn ngữ tự nhiên (cấu trúc phụ thuộc, semantic graph…). Những ứng dụng này cho thấy khả năng mô hình hóa và tổng quát hóa mạnh mẽ của GNN khi phải xử lý dữ liệu phi cấu trúc thuần túy và có quan hệ phức tạp.

Tuy nhiên, GNN vẫn đối mặt với một số thách thức như hiệu suất tính toán kém hiệu quả trên đồ thị lớn, khó khăn trong việc lựa chọn cấu trúc và siêu tham số tối ưu, cũng như vấn đề diễn giải (interpretability) của mô hình. Các hướng nghiên cứu hiện tại tập trung vào việc tối ưu hóa quy trình lan truyền thông tin, giảm thiểu chi phí tính toán thông qua sampling và clustering, cũng như phát triển các phương pháp giải thích kết quả GNN một cách minh bạch. Với sự bùng nổ về dữ liệu đồ thị và nhu cầu khai thác mối quan hệ phức tạp trong nhiều ngành, GNN hứa hẹn sẽ tiếp tục là một trong những trụ cột quan trọng trong nghiên cứu và ứng dụng của trí tuệ nhân tạo.

### 1.1. Cơ sở lý thuyết về đồ thị

Đồ thị (graph) là một đối tượng cơ bản trong toán học và khoa học máy tính, được định nghĩa dưới dạng , trong đó là tập hợp các đỉnh (nodes) và là tập hợp các cạnh (edge) nối giữa các đỉnh. Với đồ thị vô hướng (undirected), mỗi cạnh là một cặp đỉnh không thứ tự, với đồ thị có hướng (directed), mỗi cạnh được coi là một cặp đỉnh có thứ tự, có một điểm đầu và một điểm cuối. Mỗi đỉnh và cạnh đều là những thành phần cơ bản của đồ thị, và ta có thể gán thêm **trọng số** (weight) cho mỗi cạnh trong các bài toán tối ưu hóa đi.

**Thành phần cơ bản của đồ thị**

**Đỉnh (Node):** Mỗi đỉnh là một điểm cơ bản trong đồ thị. Tập hợp các đỉnh của đồ thị được ký hiệu là .

**Cạnh (Edge):** Mỗi cạnh nối hai đỉnh và là phần tử của tập . Trong đồ thị có hướng, cạnh còn được gọi là cung (arc) và có chiều cụ thể giữa một cặp đỉnh. Trong đồ thị vô hướng, cạnh được biểu diễn bằng một cặp đỉnh không có chiều. Với đồ thị khuyên (loop) thì cạnh có thể nối một đỉnh với chính nó; Với đồ thị có cạnh song song có thể có nhiều cạnh cùng nối một cặp đỉnh.

**Hướng (Directed/Undirected):** Đồ thị vô hướng là đồ thị mà các cạnh không phân biệt chiều; đồ thị có hướng thì mỗi cạnh có chiều cụ thể. Đồ thị hỗn hợp có thể chứa cả hai loại cạnh.

**Trọng số (Weight):** Trong đồ thị có trọng số, mỗi cạnh được gán một giá trị (trọng số, độ dài, chi phí,...) thể hiện tiêu chí quan tâm (như chi phí di chuyển, độ mạnh kết nối…). Đồ thị không trọng số thì có thể coi mỗi cạnh có trọng số mặc định bằng 1 hoặc chỉ quan tâm đến sự tồn tại của cạnh.

**Chu trình (Cycle) và đỉnh liền kề:** Một chu trình là một đường đi khép kín đi qua các đỉnh (bắt đầu và kết thúc tại cùng một đỉnh) mà không lặp lại cạnh. Đồ thị có chu trình gọi là đồ thị có chu trình (cyclic), ngược lại nếu không có chu trình thì đồ thị đó là **vô chu trình** (acyclic). Hai đỉnh được gọi là kề nhau (adjacent) nếu giữa chúng có một cạnh; tương tự, hai cạnh được gọi là kề nhau nếu chúng cùng gặp tại một đỉnh.

**Tính liên thông:** Đồ thị liên thông (connected) là đồ thị mà tồn tại đường đi giữa mọi cặp đỉnh bất kỳ. Nếu đồ thị không thỏa mãn điều kiện này thì gọi là rời rạc (disconnected).

**Phân loại các loại đồ thị**

**Đơn đồ thị (Simple graph):** Là đồ thị không chứa khuyên và không có cạnh song song giữa cùng một cặp đỉnh. Mỗi cặp đỉnh chỉ được nối bởi tối đa một cạnh đơn.

**Đa đồ thị (Multigraph):** Là đồ thị không thỏa mãn điều kiện đơn đồ thị, tức có thể có nhiều cạnh song song giữa một cặp đỉnh hoặc có khuyên.

**Đồ thị hai phía (Bipartite graph):** Là đồ thị mà tập đỉnh có thể phân thành hai tập con không giao nhau sao cho không có cạnh nào nối hai đỉnh trong cùng một tập. Đồ thị hai phía thường dùng để biểu diễn các mối quan hệ giữa hai nhóm đối tượng khác nhau (ví dụ: nam - nữ trong hôn nhân, khách hàng – đơn hàng trong mạng lưới giao dịch).

A purple circles with white text

Description automatically generated

**Hình 1.2:** Đồ thị vô hướng với và

### 1.2. Mô hình hoá dữ liệu đồ thị trong học máy

#### 1.2.1. Biểu diễn ma trận kề (Adjacency Matrix)

Ma trận kề là một ma trận vuông kích thước (với là số lượng đỉnh của đồ thị) dùng để biểu diễn quan hệ kề giữa các đỉnh. Mỗi phần tử trong ma trận cho biết sự kết nối giữa đỉnh và đỉnh . Theo định nghĩa, nếu có cạnh nối từ đỉnh tới đỉnh , và nếu giữa và không có cạnh nối. Trong trường hợp đồ thị vô hướng, ma trận kề sẽ đối xứng quanh đường chéo (). Nếu đồ thị có trọng số, giá trị 1 có thể được thay bằng trọng số của cạnh tương ứng. Ngược lại, với đồ thị có hướng, ma trận kề có thể không đối xứng và chỉ khi có cung từ đến .

Để xây dựng ma trận kề từ dữ liệu đồ thị, trước tiên ta đánh số các đỉnh từ 1 đến . Khởi tạo một ma trận với tất cả phần tử bằng 0. Sau đó, duyệt qua danh sách các cạnh của đồ thị. Với mỗi cạnh trong tập cạnh , ta gán (và đồng thời nếu đồ thị vô hướng). Quá trình này đảm bảo rằng những cặp đỉnh có kết nối sẽ được đánh dấu bằng 1 trong ma trận. Nếu đồ thị cho phép cạnh tự nối (self-loop) hoặc nhiều cạnh song song giữa hai đỉnh, các giá trị trên đường chéo hoặc giá trị có thể được điều chỉnh tương ứng (ví dụ: đếm số cạnh giữa và thay vì chỉ đánh dấu 1). Thông thường, đối với đồ thị đơn không có cạnh tự nối, ta giữ các phần tử đường chéo cho mọi đỉnh .

A diagram of a mathematical equation

Description automatically generated

**Hình 1.3:** Biểu diễn đồ thị bằng ma trận kề

#### 1.2.2. Biểu diễn ma trận đặc trưng đỉnh (Feature Matrix)

Bên cạnh ma trận kề mô tả cấu trúc đồ thị, mỗi đỉnh thường đi kèm một vector đặc trưng mô tả các thuộc tính hoặc thông tin liên quan đến đỉnh đó. Tập hợp các vector đặc trưng của toàn bộ đỉnh có thể được tổ chức thành một ma trận đặc trưng đỉnh kích thước , trong đó là số lượng đặc trưng mô tả mỗi đỉnh. Nói cách khác, ma trận đặc trưng đỉnh (thường ký hiệu là ) có mỗi hàng tương ứng với một đỉnh và mỗi cột tương ứng với một thuộc tính đặc trưng. Ví dụ, trong một đồ thị với đỉnh, nếu mỗi đỉnh được mô tả bởi đặc trưng, thì sẽ là ma trận chứa toàn bộ thuộc tính của các đỉnh đó. Các đặc trưng này có thể là bất kỳ thông tin gì gắn với đỉnh, tùy thuộc vào bài toán thực tế đang mô hình hóa. Việc xác định đặc trưng đỉnh phụ thuộc vào loại dữ liệu đồ thị và bài toán cần giải. Một số ví dụ điển hình gồm:

**Mạng xã hội:** Mỗi đỉnh đại diện cho một người dùng, có thể gắn các thuộc tính như tuổi, giới tính, số lượng bài đăng, số bạn bè, sở thích,... Các thuộc tính này được mã hóa thành dạng số và xếp thành vector đặc trưng cho từng người dùng. Chẳng hạn, ta có thể dùng một vector 3 chiều để biểu diễn [tuổi, số bài viết, số lượt thích] của mỗi người dùng.

**Đồ thị văn bản - tài liệu:** Xem mỗi đỉnh là một tài liệu hoặc một từ trong ngữ cảnh đồ thị ngôn ngữ. Khi đó, đặc trưng đỉnh có thể là biểu diễn nội dung của tài liệu (ví dụ: một vector độ dài cố định biểu thị tần suất các từ khóa hoặc vector word embedding của tài liệu đó).

**Đồ thị khoa học (hóa học, sinh học):** Đỉnh có thể đại diện cho phân tử hoặc nguyên tử trong đồ thị phân tử. Đặc trưng của một nguyên tử có thể bao gồm các thuộc tính hóa học như số hiệu nguyên tử, khối lượng nguyên tử, điện tích, loại nguyên tố,… Những thuộc tính này được mã hóa (ví dụ dưới dạng số thực hoặc nhị phân) và tạo thành vector đặc trưng cho mỗi nguyên tử.

Ma trận cung cấp đầu vào ban đầu về nội dung của các đỉnh cho mô hình học. Trong mạng nơ-ron đồ thị, ta không chỉ quan tâm đến cấu trúc kết nối mà còn cần tận dụng thông tin nội tại của từng đỉnh. Các thuật toán GNN sẽ học cách kết hợp đặc trưng của một đỉnh với đặc trưng của các đỉnh láng giềng (thông qua ma trận kề) để tạo ra biểu diễn mới cho đỉnh đó ở các lớp sau. Có thể hiểu, giống như dữ liệu đầu vào ban đầu cho mỗi node, và qua mỗi lớp GNN, ta thu được một ma trận đặc trưng mới (thường gọi là ma trận embedding ) với cùng số hàng nhưng số cột (chiều không gian đặc trưng) có thể thay đổi theo số lượng ẩn của lớp. Mục tiêu của quá trình này là học được biểu diễn ẩn cho mỗi đỉnh sao cho biểu diễn đó phản ánh cả thuộc tính ban đầu lẫn bối cảnh hàng xóm của đỉnh trong đồ thị. Nói một cách khác, ma trận đặc trưng đỉnh chứa thông tin định danh và thuộc tính của các thực thể (đỉnh) trong đồ thị, giúp mô hình GNN phân biệt giữa các đỉnh và cung cấp cơ sở để tổng hợp thông tin.

Ma trận kề và ma trận đặc trưng đỉnh có vai trò quan trọng trong các bài toán, các mô hình học máy liên quan đến đồ thị vì nó mã hoá cấu trúc liên kết của đồ thị, quyết định cách thức lan truyền thông tin giữa các đỉnh láng giềng. Cụ thể, ma trận kề thường được sử dụng để tổng hợp thông tin từ các đỉnh lân cận. Ví dụ, nếu là ma trận kề và là ma trận đặc trưng, thì phép nhân ma trận sẽ tạo ra một ma trận đặc trưng mới mà mỗi hàng là tổng các vector đặc trưng của những đỉnh kề với đỉnh (bao gồm cả nếu có thêm cạnh tự nối). Nói cách khác, phần tử (0 hoặc 1) khi nhân với sẽ quyết định việc đưa đặc trưng của đỉnh đóng góp vào đỉnh trong quá trình lan truyền thông tin. Nhờ đó, mô hình GNN có thể tổng hợp thông tin, mỗi đỉnh cập nhật biểu diễn của mình dựa trên đặc trưng của các đỉnh liên kết trực tiếp với nó.

Trong thực tế, để mô hình GNN học hiệu quả hơn, ma trận kề thường được điều chỉnh trước khi đưa vào thuật toán. Một kỹ thuật phổ biến là thêm cạnh tự nối (self-loop) cho mỗi đỉnh – tương đương với việc đặt cho mọi đỉnh (thêm ma trận đơn vị vào ) – nhằm bảo đảm mỗi đỉnh cũng truyền thông tin của chính nó vào quá trình tổng hợp đặc trưng. Ma trận kề cũng có thể được chuẩn hóa (normalize) để tránh trường hợp các đỉnh có nhiều láng giềng đóng góp quá lớn. Tóm lại, ma trận kề là thành phần không thể thiếu trong các thuật toán GNN, giúp mạng nơ-ron đồ thị biết được cấu trúc nào của đồ thị cần được khai thác khi cập nhật trạng thái của các đỉnh.

## 2. Học biểu diễn đồ thị (Graph Representation Learning)

Hiện nay nhiều hệ thống phức tạp có dạng đồ thị, chẳng hạn như mạng xã hội, mạng sinh học và mạng thông tin… Người ta đều thừa nhận rằng dữ liệu đồ thị thường rất phức tạp do đó khó xử lý. Để xử lý dữ liệu đồ thị một cách hiệu quả, thách thức quan trọng đầu tiên là tìm được phương pháp biểu diễn dữ liệu đồ thị hiệu quả, tức là cách biểu diễn đồ thị một cách ngắn gọn để các tác vụ phân tích nâng cao, chẳng hạn như khám phá mẫu, phân tích và dự đoán, có thể được thực hiện một cách hiệu quả cả về thời gian và không gian.

Chúng ta thường biểu diễn một đồ thị dưới dạng , trong đó là tập các đỉnh (node) và là tập các cạnh (edge). Đới với các đồ thị lớn, chẳng hạn như đồ thị có hàng tỷ đỉnh, phương pháp biểu diễn truyền thống đặt ra nhiều thách thức cho việc xử lý và phân tích đồ thị.

- Độ phức tạp tính toán cao:Các mối quan hệ được mã hoá bởi tập cạnh đòi hỏi hầu hết các thuật toán xử lý hay phân tích đồ thị phải thực hiện các bước tính toán lặp hoặc tổ hợp. Ví dụ, một cách phổ biến là sử dụng độ dài đường đi ngắn nhất hoặc trung bình giữa hai đỉnh để biểu diễn khoảng cách của chúng. Để tính khoảng cách này theo biểu diễn truyền thống, ta phải liệt kê rất nhiều đường đi khả dĩ giữa hai đỉnh, vốn về bản chất là một bài toán tổ hợp. Các phương pháp như vậy dẫn đến độ phức tạp tính toán cao, khiến chúng không thể áp dụng cho các đồ thị quy mô lớn trong thực tế.

- **Khả năng song song hoá thấp:** Tính toán song song và phân tán đã trở thành phương pháp mặc định để xử lý và phân tích dữ liệu quy mô lớn. Tuy nhiên, dữ liệu đồ thị khi được biểu diễn theo cách truyền thống lại gặp khó khăn lớn trong việc thiết kế và triển khai các thuật toán song song và phân tán. Đỉnh trong đồ thị bị ràng buộc với nhau thông qua , do đó khi phân tán các tập con đỉnh sang các phân vùng hoặc máy chủ khác nhau, chi phí giao tiếp cao giữa các máy chủ trở nên rất nặng nề và làm hạn chế tỉ lệ tăng tốc.

- **Không khả dụng cho các phương pháp học máy:** Gần đây, các phương pháp học máy, đặc biệt là học sâu, tỏ ra rất mạnh mẽ ở nhiều lĩnh vực. Tuy nhiên, khi dữ liệu đồ thị được biểu diễn theo cách truyền thống, hầu hết các phương pháp học máy có sẵn không thể áp dụng được. Những phương pháp này thường giả định rằng các mẫu dữ liệu có thể được biểu diễn như các vector độc lập trong một không gian vector, trong khi các mẫu trong dữ liệu đồ thị (tức là các đỉnh) lại phụ thuộc lẫn nhau ở mức độ nhất định do xác định. Mặc dù ta có thể đơn giản biểu diễn một đỉnh bằng vector hàng tương ứng trong ma trận kề của đồ thị, song kích thước cực lớn của vector đó trong đồ thị có nhiều đỉnh khiến việc xử lý và phân tích đồ thị sau đó trở nên khó khăn.

Để giải quyết những thách thức này, cần phát triển phương pháp học biểu diễn đồ thị mới, tức là học các biểu diễn vectơ chiều thấp liên tục và dày đặc cho các đỉnh, để có thể giảm nhiễu hoặc thông tin dư thừa và bảo toàn thông tin cấu trúc nội tại. Trong không gian biểu diễn đã học, các mối quan hệ giữa các đỉnh, ban đầu được biểu diễn bằng các cạnh hoặc các phép đo tôpô bậc cao khác trong đồ thị, được nắm bắt bằng khoảng cách giữa các đỉnh trong không gian vectơ và các đặc điểm cấu trúc của một đỉnh được mã hóa vào vectơ biểu diễn của nó. Về cơ bản, để không gian biểu diễn có thể hỗ trợ tốt các tác vụ phân tích đồ thị, học biểu diễn đồ thị cần đạt hai mục tiêu:

- **Phục hồi lại đồ thị gốc:** Nếu có một cạnh hoặc quan hệ giữa hai đỉnh, thì khoảng cách giữa hai vector biểu diễn tương ứng phải tương đối nhỏ, sao cho đồ thị ban đầu có thể được khôi phục từ không gian biểu diễn.

- Hỗ trợ hiệu quả suy luận trên đồ thị: Không gian biểu diễn phải giúp dự đoán các cạnh chưa biết, xác định các đỉnh quan trọng, và suy luận nhãn đỉnh. Chỉ đạt mục tiêu khôi phục đồ thị thôi chưa đủ để hỗ trợ suy luận. Sau khi có biểu diễn, các tác vụ mức thấp như phân loại đỉnh, phân cụm đỉnh, trực quan hóa đồ thị và dự đoán liên kết có thể được giải quyết dựa trên các biểu diễn này.

### 2.1. Biểu diễn đồ thị truyền thống

Các phương pháp biểu diễn đồ thị truyền thống ban đầu được nghiên cứu như các kỹ thuật giảm chiều. Một đồ thị thường được xây dựng từ một tập dữ liệu được biểu diễn bởi các đặc trưng. Như đã đề cập trước đó, biểu diễn đồ thị thường có hai mục tiêu, đó là tái tạo cấu trúc đồ thị gốc và hỗ trợ suy luận trên đồ thị. Các hàm mục tiêu của phương pháp biêu diễn đồ thị truyền thống chủ yếu hướng tới mục tiêu tái tạo đồ thị.

Cụ thể, đầu tiên xây dựng một đồ thị lân cận bằng các thuật toán kết nối như K láng giềng gần nhất (K Nearest Neighbors - KNN). Tiếp theo, dựa trên , có thể tính toán đường đi ngắn nhất giữa các điểm dữ liệu khác nhau. Kết quả là, đối với tất cả N điểm dữ liệu trong tập, ta có ma trận khoảng cách trên đồ thị. Cuối cùng, phương pháp đa chiều cổ điển (Multidimensional Scaling - MDS) được áp dụng lên ma trận này để thu được các vector tọa độ. Các biểu diễn mà Isometric Mapping (Isomap) học được gần đúng bảo toàn khoảng cách giữa các cặp điểm trong không gian có chiều thấp. Vấn đề then chốt của Isomap là độ phức tạp cao do việc tính toán đường đi ngắn nhất cho từng cặp điểm.

Phương pháp Locally Linear Embedding (LLE) được đề xuất nhằm loại bỏ việc phải ước tính khoảng cách giữa các cặp điểm cách xa nhau. LLE giả định rằng mỗi điểm và các điểm lân cận của nó nằm trên hoặc gần một vùng tuyến tính cục bộ của đa tạp. Để đặc trưng hóa hình học cục bộ, mỗi điểm có thể được tái tạo từ các điểm lân cận. Cuối cùng, trong không gian có chiều thấp, LLE xây dựng một ánh xạ bảo toàn lân cận dựa trên tái tạo tuyến tính cục bộ.

Trong khi đó, phương pháp Laplacian Eigenmaps (LE) cũng bắt đầu với việc xây dựng đồ thị sử dụng vùng lân cận hoặc KNN. Tiếp theo, hàm Heat Kernel được sử dụng xác định trọng số giữa hai đỉnh trong đồ thị. Cuối cùng, các biểu diễn đỉnh có thể thu được bằng cách dựa trên quy tắc hóa ma trận Laplacian. Hơn nữa, phép chiếu bảo toàn tính cục bộ (Locality Preserving Projection - LPP) một xấp xỉ tuyến tính của LE phi tuyến, cũng được đề xuất.

### 2.2. Mạng nơ-ron trên đồ thị

Trong thập kỷ qua, học sâu đã trở thành “viên ngọc quý” của trí tuệ nhân tạo và học máy, cho thấy hiệu suất vượt trội trong âm thanh, hình ảnh và xử lý ngôn ngữ tự nhiên,... Mặc dù rõ ràng là đồ thị tồn tại khắp nơi trong thế giới thực, song việc ứng dụng các phương pháp học sâu để phân tích dữ liệu đồ thị là rất khó. Vấn đề này không hề đơn giản vì các thách thức sau đây:

- **Cấu trúc không đều của đồ thị:** Không giống như hình ảnh, âm thanh và văn bản có cấu trúc lưới rõ ràng, đồ thị có cấu trúc không đều, làm cho việc tổng quát hóa một số phép toán cơ bản lên đồ thị trở nên khó khăn. Ví dụ, định nghĩa các phép tích chập và gộp, vốn là các phép toán nền tảng trong mạng nơ-ron tích chập (CNN), đối với dữ liệu đồ thị là không đơn giản.

- **Tính dị hướng và đa dạng của đồ thị**: Một đồ thị có thể rất phức tạp, chứa nhiêu loại và tính chất khác nhau. Những dạng, tính chất và nhiệm vụ đa dạng này đòi hỏi các kiến trúc mô hình khác nhau để giải quyết các bài toán cụ thể.

- **Đồ thị có quy mô lớn**: Trong kỷ nguyên dữ liệu lớn, đồ thị thực có thể có hàng triệu hoặc hàng tỷ đỉnh và cạnh. Thiết kế các mô hình có khả năng mở rộng, ưu tiên các mô hình có độ phức tạp thời gian tuyến tính theo kích thước đồ thị, là vấn đề then chốt.

- **Kết hợp kiến thức liên ngành:** Đồ thị thường liên quan đến các ngành khác nhau như sinh học, hóa học và khoa học xã hội… Bản chất liên ngành này đồng thời mang lại cơ hội và thách thức. Kiến thức chuyên môn có thể được tận dụng để giải quyết các bài toán cụ thể nhưng việc tích hợp kiến thức chuyên ngành làm cho thiết kế mô hình trở nên phức tạp.

Hiện nay, mạng nơ-ron trên đồ thị đã thu hút được sự quan tâm đáng kể của giới nghiên cứu trong vài năm gần đây. Các kiến trúc được áp dụng và chiến lược huấn luyện rất đa dạng, từ giám sát đến không giám sát và từ mô hình tích chập đến mô hình hồi quy, bao gồm các mạng nơ-ron hồi quy trên đồ thị (Graph RNN), mạng nơ-ron tích chập trên đồ thị (GCN), bộ mã hóa tự động trên đồ thị (GAE), học tăng cường trên đồ thị (Graph RL) và các phương pháp đối kháng trên đồ thị. Có nhiều hướng nghiên cứu đang diễn ra hoặc dự kiến trong tương lai cũng rất đáng được tiếp tục nghiên cứu, bao gồm các mô hình mới cho các cấu trúc đồ thị chưa được khảo sát, đồ thị động, khả năng giải thích và tính bền vững,... Nhìn chung, học sâu trên đồ thị là một lĩnh vực nghiên cứu đầy hứa hẹn và đang phát triển nhanh chóng, vừa mang đến những cơ hội vừa đặt ra nhiều thách thức. Việc nghiên cứu học sâu trên đồ thị là một bước xây dựng then chốt trong mô hình hoá dữ liệu quan hệ và là một bước quan trọng hướng tới tương lai với các kỹ thuật học máy và trí tuệ nhân tạo ngày càng tốt hơn.

## 3. Ứng dụng của GNN trong các lĩnh vực

A group of icons with text

Description automatically generated with medium confidence

**Hình 1.4:** Các bài toán điển hình trong học sâu trên GNN

### 3.1. Mạng xã hội và phân tích người dùng

#### 3.1.1. Phân tích tài khoản thật – giả (Real vs. Fake Account Detection)

Phân tích tài khoản thật – giả trên mạng xã hội nhằm xác định và phân loại các tài khoản thật (real) và tài khoản ảo (fake) do con người hoặc bot tạo ra. Trong đồ thị người dùng, mỗi đỉnh đại diện cho một tài khoản, các cạnh biểu diễn mối quan hệ như tương tác, theo dõi hoặc bình luận. Bài toán phát hiện tài khoản thật – giả tìm cách học một hàm ánh xạ các đỉnh vào không gian sao cho tài khoản thật và tài khoản giả được phân tách rõ ràng về vị trí. GNN tận dụng cơ chế truyền tin (message passing) giữa các đỉnh láng giềng để thu thập thông tin về hành vi kết nối và đặc trưng nội tại (profile metadata, lịch sử đăng bài, tần suất tương tác…). Qua mỗi lớp truyền tin, GNN không chỉ học được biểu diễn cục bộ dựa trên đặc trưng của đỉnh mà còn khái quát hóa cấu trúc xung quanh, từ đó khuếch đại khác biệt giữa các nhóm tài khoản thật và tài khoản giả. Nhiều nghiên cứu đã áp dụng GNN cho bài toán này, bao gồm cả phương pháp học có giám sát và không giám sát. Kết quả cho thấy GNN hiệu quả cao trong việc giảm tỉ lệ báo cáo sai (false positives) và bỏ sót (false negatives) so với các phương pháp truyền thống chỉ dựa vào đặc trưng đơn luồng (feature-based).

#### 3.1.2. Phân tích cộng đồng (Community Detection)

Phântích cộng đồng trên mạng xã hội nhằm xác định các nhóm người dùng có mức độ kết nối mật thiết với nhau. Ví dụ, các nhóm bạn bè hay nhóm người dùng có chung sở thích thường tạo thành những cộng đồng. Bài toán cộng đồng tìm cách phân loại các đỉnh trong đồ thị thành các nhóm sao cho bên trong nhóm có nhiều cạnh hơn so với giữa các nhóm. GNN có khả năng học biểu diễn cho từng đỉnh dựa trên cấu trúc lân cận, giúp các đỉnh cùng cộng đồng có biểu diễn tiệm cận nhau. Qua các lớp truyền tin, GNN thu thập thông tin về kết nối và tầng lớp xung quanh, từ đó khuếch đại sự tương đồng của các đỉnh trong cùng một cộng đồng. Các nghiên cứu ghi nhận GNN đã được áp dụng thành công cho bài toán phát hiện cộng đồng, bao gồm cả trường hợp học có giám sát (đã biết một số cấu trúc cộng đồng) và không giám sát.

#### 3.1.3. Dự đoán liên kết (Link Prediction)

Bài toán dự đoán liên kết đặt mục tiêu tìm các cạnh tiềm năng giữa hai đỉnh chưa có kết nối hiện tại. Ví dụ, tính năng “Gợi ý kết bạn” (friend recommendation) trên Facebook, Twitter, Zalo… dự đoán người dùng nào có khả năng kết bạn với nhau dựa trên mạng lưới hiện tại. Tương tự, trong hệ thống đề xuất nội dung, ta có thể xem bài toán gợi ý bạn bè là một dạng đề bài dự đoán liên kết trong đồ thị người dùng. GNN học được biểu diễn ngầm của người dùng từ cấu trúc mạng xã hội, sau đó sử dụng biểu diễn này để đánh giá khả năng tồn tại của một liên kết mới. Thông qua quá trình truyền tin, GNN tự động tổng hợp thông tin của các đỉnh lân cận, nên các đỉnh có kết nối gián tiếp với nhau nhiều khả năng sẽ có biểu diễn gần nhau và được dự đoán thành liên kết. Điều này cho phép GNN nắm bắt ảnh hưởng xã hội và thông tin ngữ cảnh rộng hơn thay vì chỉ dựa trên số lượng bạn chung hay tiêu chí cục bộ. Các nghiên cứu cho thấy GNN có tiềm năng lớn trong học biểu diễn người dùng trên mạng xã hội nhờ cơ chế truyền tin theo lân cận, từ đó hỗ trợ tốt bài toán gợi ý bạn bè.

### 3.2. Phân loại và phát hiện bất thường

#### 3.2.1. Phân loại đồ thị sinh học

Trong sinh học và hóa học, nhiều đối tượng được biểu diễn dưới dạng đồ thị (ví dụ phân tử hóa học, mạng tương tác protein…). Bài toán phân loại đồ thị (graph classification) là gán nhãn cho cả đồ thị, chẳng hạn phân loại phân tử theo tính chất vật lý hoặc hoạt tính sinh học (độc tính, khả năng tương tác thuốc…). Mỗi phân tử có thể coi là một đồ thị với các đỉnh là nguyên tử và các cạnh là liên kết hóa học. Ví dụ, bộ dữ liệu MUTAG gồm 188 phân tử nitroaromatic được sử dụng để dự đoán tính đột biến của phân tử. GNN được thiết kế để tận dụng cấu trúc đồ thị và truyền tin giữa các đỉnh kề nhau, do đó có thể học được đặc trưng sâu sắc từ cấu trúc topology của phân tử. Các nghiên cứu chỉ ra GNN thể hiện hiệu quả cao trong dự đoán tính chất phân tử nhờ khai thác thông tin cấu trúc đồ thị. So với phương pháp truyền thống bỏ qua cấu trúc liên kết, GNN có ưu thế trong việc kết hợp đồng thời thông tin thuộc tính đỉnh và liên kết.

#### 3.2.2. Phát hiện gian lận tài chính

Trong lĩnh vực tài chính, bài toán phát hiện gian lận (fraud detection) thường liên quan đến mạng lưới giao dịch giữa các tài khoản. Ví dụ, phát hiện giao dịch thẻ tín dụng gian lận, rửa tiền, hoặc các bất thường khác trong mạng lưới giao dịch ngân hàng… Mỗi tài khoản và giao dịch có thể được mô hình hóa như một đỉnh và một cạnh trong đồ thị giao dịch, và “gian lận” biểu hiện thành các mô hình kết nối phức tạp trong đồ thị. GNN rất phù hợp vì nó xử lý trực tiếp dữ liệu dạng đồ thị. Thay vì phân tích từng giao dịch riêng lẻ, GNN có thể xem xét mối quan hệ giữa các giao dịch thông qua đồ thị chung. Cụ thể, GNN coi các tài khoản, giao dịch, thiết bị như các đỉnh liên kết với nhau, nhờ đó có thể phát hiện các mô hình gian lận tiềm ẩn trong toàn bộ mạng lưới. Các phương pháp truyền thống (như XGBoost) chỉ phân tích giao dịch đơn lẻ, trong khi GNN tận dụng ngữ cảnh đồ thị để giảm báo động sai và tăng độ chính xác phát hiện.

**3.3. Một số ứng dụng khác**

**Xử lý tài liệu và ngôn ngữ:** tài liệu – ngôn ngữ cũng có thể được mô hình hóa dưới dạng đồ thị, trong đó các đỉnh biểu diễn các thực thể như từ, cụm từ, câu hoặc đoạn văn, còn các cạnh thể hiện mối quan hệ ngữ nghĩa, cú pháp hay tham chiếu giữa chúng. GNN đã được áp dụng rộng rãi trong các hệ thống xử lý ngôn ngữ tự nhiên (Natural Language Processing – NLP) tiên tiến để giải quyết những bài toán như phân loại văn bản, trích xuất quan hệ, sinh tóm tắt tự động, hỏi – đáp và hoàn thiện ngữ cảnh. Nhờ khả năng xử lý hiệu quả cấu trúc đồ thị phức tạp, các mô hình GNN thường đạt hiệu quả hàng đầu trong việc khai thác mối liên kết ngữ nghĩa giữa các thành tố của văn bản. Các mô hình GNN trong NLP tận dụng thông tin lịch sử (ngữ cảnh tuần tự trong câu) và mối liên hệ không gian (đường dẫn ngữ nghĩa giữa các thực thể) để nâng cao độ chính xác cho bài toán. Khi phân loại tài liệu, GNN có thể lan truyền thông tin giữa các đỉnh từ khóa để nhận diện chủ đề. Khi trích xuất quan hệ, GNN sử dụng đường đi ngắn nhất trên đồ thị phụ thuộc cú pháp để xác định chính xác mối quan hệ giữa hai thực thể. Khi tóm tắt, GNN kết hợp cấu trúc câu với trọng số quan trọng để sinh đoạn văn ngắn gọn mà vẫn đầy đủ ý chính. Nhờ đó, các ứng dụng NLP tích hợp GNN góp phần nâng cao chất lượng hiểu và sinh ngôn ngữ, đáp ứng tốt các yêu cầu của các hệ thống xử lý ngôn ngữ tự nhiên hiện đại.

**Giao thông:** Mạng lưới giao thông (đường bộ, đường sắt, phương tiện giao thông…) cũng có thể biểu diễn thành đồ thị không gian - thời gian. GNN được áp dụng rộng rãi trong các hệ thống giao thông thông minh để dự báo lưu lượng, điều khiển tín hiệu và quản lý giao thông. Nhờ xử lý được thông tin không gian phức tạp, GNN đã đạt hiệu quả hàng đầu trong dự báo tình trạng giao thông, ví dụ dự báo lưu lượng và tốc độ dòng xe, dự báo lưu lượng hành khách trên mạng lưới đường sắt đô thị, cũng như nhu cầu gọi xe công cộng. Các mô hình GNN khai thác lịch sử và mối liên hệ giữa các nút giao thông để cung cấp dự báo chính xác, giúp giảm ùn tắc và tối ưu hóa quản lý giao thông.

**Năng lượng:** Trong ngành năng lượng, lưới điện phân phối và truyền tải cũng tạo thành một đồ thị (các trạm biến áp, máy phát, đường dây). GNN được dùng để dự báo công suất tiêu thụ, cân bằng tải và phát hiện sự cố mạng lưới. Ví dụ, một nghiên cứu trên dữ liệu lưới điện Bắc Âu ứng dụng GNN (GCN kết hợp GRU và biến thể Fourier GNN) để dự báo vị trí công suất, cho hiệu năng cạnh tranh với các mô hình dự báo truyền thống. Điều này cho thấy GNN tận dụng tốt cấu trúc đồ thị vốn có của lưới điện để nâng cao độ chính xác dự báo và giám sát hệ thống năng lượng.

Ngoài ra, GNN còn được ứng dụng trong nhiều lĩnh vực khác như IoT, mạng lưới cảm biến, khuyến nghị sản phẩm và các vấn đề khoa học khác liên quan mạng phức tạp… Tóm lại, GNN là công cụ mạnh mẽ cho các bài toán liên quan dữ liệu đồ thị nhờ khả năng trích xuất thông tin cấu trúc và thuộc tính sâu rộng của đồ thị trong nhiều miền ứng dụng khác nhau.

# Chương 2: Phân tích các mô hình GNN phù hợp với bài toán phân loại tài khoản mạng xã hội.

## 1. Đặt vấn đề

Trong kỷ nguyên số hiện nay, mạng xã hội đóng vai trò trung tâm trong việc kết nối, chia sẻ thông tin và tương tác giữa người dùng trên toàn cầu. Tuy nhiên, chính môi trường mở và phân tán này cũng tạo ra “mảnh đất màu mỡ” cho hoạt động của các tài khoản ảo (fake accounts) với nhiều mục đích tiêu cực như phát tán tin giả, quảng bá thông tin sai lệch, gây nhiễu loạn dư luận hoặc thực hiện các hành vi lừa đảo. Việc phân biệt giữa tài khoản thật (real) và tài khoản giả (fake) không chỉ là thách thức kỹ thuật mà còn là yêu cầu cấp bách để bảo đảm tính an toàn, tin cậy cho cộng đồng người dùng mạng xã hội.

Các phương pháp phát hiện tài khoản giả truyền thống thường dựa trên các đặc trưng thủ công như tần suất đăng bài, lịch sử tương tác, số lượng bạn bè, độ dài văn bản, hay các chỉ số hành vi cơ bản. Tuy nhiên, những phương pháp này thường bỏ qua mối quan hệ phức tạp giữa người dùng và các yếu tố ngữ cảnh trong mạng lưới xã hội. Trong khi đó, mạng xã hội thực chất là một đồ thị lớn, nơi mỗi người dùng là một đỉnh (node), các kết nối giữa họ là các cạnh (edge), và thông tin lan truyền qua từng cạnh mang theo đặc trưng hành vi, sở thích, nhóm cộng đồng... Chính cấu trúc đồ thị này cung cấp các tín hiệu quan trọng để phân biệt tài khoản thật và giả, đặc biệt khi kẻ tạo tài khoản ảo thường cố gắng sao chép hành vi “bề ngoài” nhưng khó tái hiện chính xác các mô hình tương tác phức hợp trong mạng.

GNN ra đời như một giải pháp tối ưu để khai thác triệt để thông tin cấu trúc và ngữ nghĩa trên đồ thị. Qua cơ chế “message passing” – truyền và tổng hợp thông tin giữa các đỉnh lân cận – GNN có khả năng học được biểu diễn (embedding) giàu ý nghĩa cho từng đỉnh, phản ánh không chỉ đặc trưng riêng lẻ mà còn mối quan hệ đa chiều với cộng đồng xung quanh. Nhờ vậy, GNN hứa hẹn nâng cao độ chính xác trong việc phát hiện tài khoản giả so với các phương pháp chỉ dựa trên đặc trưng cục bộ.

Tuy nhiên, việc ứng dụng GNN cho bài toán phân loại tài khoản mạng xã hội cũng đặt ra nhiều thách thức. Thứ nhất, quy mô đồ thị mạng xã hội thường rất lớn, với hàng triệu đến hàng trăm triệu đỉnh và cạnh, đòi hỏi mô hình phải xử lý hiệu quả về mặt tính toán và bộ nhớ. Thứ hai, đặc trưng của tài khoản giả có thể rất đa dạng như bot, tự động đăng bài, tài khoản do con người điều hành phục vụ mục đích tấn công tâm lý, mạng lưới tài khoản giả tương tác lẫn nhau.... Điều này yêu cầu GNN phải chịu được sự đa hình của loại đồ thị – bao gồm cả đồ thị đơn loại (homogeneous graph) và đồ thị hỗn hợp (heterogeneous graph). Thứ ba, giữa việc tối ưu hóa hiệu suất phân loại và khả năng giải thích (interpretability) của mô hình cũng tồn tại mâu thuẫn. Càng nhiều tầng sâu và hàm chú ý phức tạp, càng khó lý giải quyết định của GNN, trong khi việc truy xuất nguồn gốc kết quả có ý nghĩa quan trọng trong bối cảnh phòng chống tin giả.

Vì vậy, Chương 2 sẽ tập trung phân tích và so sánh các kiến trúc GNN tiêu biểu bao gồm Graph Convolutional Network (GCN), Graph Attention Network (GAT), GraphSAGE cũng như các biến thể hỗn hợp và đa chiều để đánh giá mức độ phù hợp của chúng với bài toán phân loại tài khoản mạng xã hội. Đồng thời, xem xét các kỹ thuật tối ưu hóa như sampling, clustering, hierarchical pooling để giải quyết bài toán quy mô lớn, và các phương pháp giải thích model nhằm tăng cường tính minh bạch. Mục tiêu cuối cùng là lựa chọn và đề xuất một mô hình GNN có hiệu quả cao, khả năng mở rộng tốt, đồng thời đảm bảo tính giải thích đủ để ứng dụng thực tiễn trong hệ thống phát hiện tài khoản giả trên các nền tảng mạng xã hội.

## 2. Một số kiến trúc của GNN

Ý tưởng cốt lõi của mạng nơ-ron đồ thị là cập nhật tuần tự (lần lượt) các biểu diễn của đỉnh bằng cách kết hợp biểu diễn của các đỉnh lân cận với biểu diễn của chính nó. Bắt đầu từ biểu diễn đỉnh ban đầu , ở mỗi lớp chúng ta có hai hàm quan trọng:

- **Aggregate**: hàm này cố gắng tập hợp thông tin từ các đỉnh lân cận của mỗi đỉnh.

- **Combine**: hàm này cố gấng cập nhật biêu diễn của đỉnh bằng cách kết hợp thông tin đã tập hợp từ các đỉnh lân cận với biểu diễn hiện tại của đỉnh.

Về mặt toán học, chúng ta có thể định nghĩa khung tổng quát của mạng nơ-ron đồ thị như sau:

Khởi tạo:

Với

với là tập các đỉnh lân cận của đỉnh thứ . Các biểu diễn đỉnh ở lớp cuối cùng có thể được xem như biểu diễn cuối cùng của các đỉnh.

Khi đã có biểu diễn của các đỉnh, chúng có thể được sử dụng cho các tác vụ downstream. Ví dụ phân loại đỉnh, nhãn của đỉnh (ký hiệu ) có thể được dự đoán thông qua hàm Softmax, tức là

với , và là số nhãn trong không gian đầu ra.

Với một tập các đỉnh đã được gán nhān, toàn bộ mô hình có thể được huấn luyện bằng cách cực tiểu hóa hàm mất mát sau:

với là nhãn thật của đỉnh là số lượng đỉnh đã được gán nhãn, loss là một hàm mất mát như hàm mất mát cross-entropy. Toàn bộ mạng nơ-ron có thể được tối ưu bằng cách thu nhỏ giá trị thông qua thuật toán lan truyền ngược (backpropagation).

### 2.1. Graph Convolutional Network - GCN

Mạng tích chập đồ thị (GCN - Graph Convolutional Network) hiện là kiến trúc mạng đồ thị phổ biến nhất nhờ tính đơn giản và hiệu quả trong nhiều tác vụ và ứng dụng khác nhau. Cụ thể, biểu diễn của các đỉnh ở mỗi lớp được cập nhật theo quy tắc truyền dẫn sau:

với:

- : là ma trận kề của đồ thị vô hướng đã cộng thêm các tự kết nối (self-loops), cho phép kết hợp luôn đặc trưng của chính đỉnh đó khi cập nhật biểu diễn đỉnh.

- : là ma trận đơn vị.

- : là ma trận đường chéo với .

- **Hàm**  : là hàm kích hoạt như ReLU hoặc Tanh. Hàm ReLU thường dùng là .

- **Ma trận** : là tham số tuyến tính sẽ được huấn luyện trong quá trình tối ưu.

A diagram of a network

Description automatically generated

**Hình 2.1:** Graph Convolutional Network

Ta có thể bóc tách phương trình (2.5) để thấy hai thành phần Aggregate và Combine trong GCN. Với một đỉnh , phương trình cập nhật biểu diễn đỉnh có thể viết lại như sau:

Trong (2.7) ta thấy hàm Aggregate được định nghĩa như trung bình các trọng số của các đỉnh lân cận. Trọng số của đỉnh lân cận được xác định bởi trọng số của cạnh nối giữa và (tức là đã được chuẩn hóa theo bậc của cả hai đỉnh). Hàm Combine được định nghĩa là tổng của các thông điệp đã được tập hợp và chính biểu diễn của đỉnh, trong đó biểu diễn đỉnh được chuẩn hóa theo bậc của nó.

### 2.2. Graph Attention Network – GAT

Trong GCN, đối với một đỉnh mục tiêu , tầm quan trọng của một đỉnh lân cận được xác định bởi trọng số của cạnh (được chuẩn hóa theo bậc của các đỉnh). Tuy nhiên, trong thực tế, đồ thị đầu vào có thể bị nhiễu. Trọng số các cạnh có thể không phản ánh đúng độ mạnh thực sự giữa hai đỉnh. Do đó, một phương pháp có cơ sở vững chắc hơn là tự động học tầm quan trọng của từng đỉnh lân cận. Graph Attention Network (GAT) được xây dựng dựa trên ý tưởng này và cố gắng học tầm quan trọng của mỗi đỉnh lân cận dựa trên cơ chế Attention. Cơ chế Attention đã được sử dụng rộng rãi trong nhiều tác vụ về ngôn ngữ tự nhiên và thị giác máy tính.

**Graph Attention Layer:** định nghĩa cách chuyển các biểu diễn ẩn của các đỉnh tại lớp (ký hiệu là ) sang các biểu diễn đỉnh mới . Để đảm bảo đủ khả năng biểu đạt khi biến đổi các biểu diễn đỉnh cấp thấp lên cấp cao hơn, một phép biến đổi tuyến tính chung được áp dụng cho mọi đỉnh, ký hiệu là . Sau đó, self-attention được định nghĩa trên các đỉnh, đo lường hệ số attention cho bất kỳ cặp đỉnh nào thông qua một cơ chế attention chung :

với biểu thị độ mạnh quan hệ giữa đỉnh và . Lưu ý trong phần này dùng để biểu diễn một vector theo cột thay vì vector theo hàng. Vê lý thuyết, với mỗi đỉnh ta có thể cho nó “chú ý” đến mọi đỉnh khác trên đồ thị, nhưng điều này lại bỏ qua thông tin cấu trúc của đồ thị. Giải pháp hợp lý hơn là chỉ chú ý đến các đỉnh lân cận của mỗi đỉnh. Trong thực tế, chỉ các đỉnh bậc nhất (bao gồm cả chính nó) được sử dụng. Để các hệ số có thể so sánh được giữa các đỉnh khác nhau, các hệ số attention thường được chuẩn hóa bằng hàm softmax:

Ta thấy rằng với một đỉnh về cơ bản định nghĩa một phân phối đa thức trên các đỉnh lân cận, cũng có thể hiểu như xác suất chuyển tiếp từ tới từng đỉnh lân cận của nó. Cơ chế attention được định nghĩa như một mạng nơ-ron truyền thẳng một lớp gồm một phép biến đổi tuyến tính với vector trọng số và hàm kích hoạt phi tuyến LeakyReLU (với hệ số dốc âm ). Cụ thể, ta tính hệ số attention theo kiến trúc:

với || biểu thị phép ghép nối hai vector. Biểu diễn đỉnh mới là một tổ hợp tuyến tính của các vector lân cận với trọng số là các hệ số attention (có thể là một biến đổi phi tuyến), tức là:

**Multi-head Attention:** Trong thực tế, thay vì chỉ sử dụng một cơ chế attention duy nhất, ta có thể dùng cơ chế multi-head attention, trong đó mỗi head sẽ xác định một hàm đo độ tương đồng khác nhau giữa các đỉnh. Đối với mỗi attention head, ta độc lập thu được một biểu diến đỉnh mới theo phương trình (2.11). Biểu diễn cuối cùng của mỗi đỉnh sẽ là phép ghép nối (concatenation) các biểu diễn đỉnh do từng attention head học được. Về mặt toán học, ta có:

với là tổng số attention head, là hệ số attention được tính từ head thứ , và là ma trận biến đổi tuyến tính của head thứ . Ở lớp cuối cùng, khi cố gắng kết hợp các biểu diễn đỉnh từ các attention head khác nhau, thay vì dùng phép ghép nối (concatenation), có thể sử dụng các kỹ thuật pooling khác như chỉ cần lấy trung bình các biểu diễn đỉnh từ các attention head khác nhau. Khi đó:

### 2.3. Graph Sample and Aggregate – GraphSAGE

Ở giai đoạn đầu của sự phát triển GNN, phần lớn các nghiên cứu tập trung vào học chuyển tiếp (transductive learning) trên đồ thị có kích thước cố định, trong khi phương pháp học quy nạp (inductive learning) lại thực tiễn hơn trong nhiều trường hợp. GraphSAGE là một mô hình học biểu diễn (embedding) cho các đỉnh trong đồ thị theo kiểu quy nạp, tức là có thể tổng quát hóa để biểu diễn các đỉnh mới chưa từng xuất hiện trong quá trình huấn luyện. Kiến trúc tổng thể của GraphSAGE được minh họa trong Hình 6.

A diagram of a network

Description automatically generated

**Hình 2.2:** Kiến trúc tổng thể của GraphSAGE

GraphSAGE có thể được xem là một phần mở rộng của GNN. Sự mở rộng đầu tiên là hàm tổng hợp (aggregator) được khái quát hóa. Cho đồ thị là tập lân cận của đỉnh là biểu diễn của đỉnh, thì quá trình tạo biểu diễn (embedding) tại lớp của đỉnh có thể được biểu diễn như sau:

khác với hàm tổng hợp trung bình gốc trong GCN, GraphSAGE đề xuất các bộ tổng hợp LSTM và Pooling để tổng hợp thông tin từ các đỉnh lân cận.

Sự mở rộng thứ hai là GraphSAGE áp dụng hàm nối (concatenation) để kết hợp thông tin của đỉnh mục tiêu và các đỉnh lân cận thay vì hàm tổng hợp như sau:

trong đó là ma trận trọng số, và là hàm kích hoạt.

Để làm cho GNN phù hợp với các đồ thị lớn, GraphSAGE giới thiệu chiến lược huấn luyện mini-batch nhằm giảm chi phí tính toán trong giai đoạn huấn luyện. Cụ thể, trong mỗi vòng huấn luyện, chỉ các đỉnh được sử dụng để tính toán biểu diển trong batch mới được xét đến, điều này giúp giảm đáng kể số lượng đỉnh được lấy mẫu. Lấy lớp 2 trong Hình 7(a) làm ví dụ. Không giống như huấn luyện toàn phần (full-batch training), vốn xét đến toàn bộ 11 đỉnh, chỉ có 6 đỉnh được sử dụng cho huấn luyện mini-batch. Tuy nhiên, cách triển khai đơn giản của huấn luyện mini-batch gặp phải vấn đề mở rộng lân cận (neighborhood expansion problem). Như minh họa ở lớp 1 của Hình 6.4(a), phần lớn các đỉnh được lấy mẫu bởi vì số lượng đỉnh lân cận tăng theo cấp số nhân nếu tất cả các đỉnh lân cận đều được lấy mẫu ở mỗi lớp. Do đó, tất cả các đỉnh cuối cùng sẽ được chọn nếu mô hình có nhiêu lớp.

A diagram of a fixed-size neighbor

Description automatically generated

**Hình 2.3:** So sánh giữa mini-batch training và lấy mẫu lân cận kích thước cố định

Để cải thiện hiệu quả đào tạo và loại bỏ vấn đề mở rộng lân cận, GraphSAGE áp dụng chiến lược lấy mẫu lân cận có kích thước cố định. Cụ thể, một tập hợp các đỉnh lân cận có kích thước cố định được lấy mẫu cho mỗi lớp để tính toán, thay vì sử dụng toàn bộ các tập hợp đỉnh lân cận. Ví dụ, người ta có thể đặt tập hợp có kích thước cố định là hai đỉnh lân cận, được minh họa trong Hình 7(b), các đỉnh màu vàng biểu diễn các đỉnh được lấy mẫu và các đỉnh màu xanh là các đỉnh ứng viên. Người ta quan sát thấy rằng số lượng các đỉnh được lấy mẫu giảm đáng kể, đặc biệt là đối với lớp 1.

Tóm lại, GraphSAGE là phương pháp học biểu diễn theo hướng quy nạp (inductive learning) trên các đồ thị lớn. Phương pháp này giới thiệu một hàm tổng hợp được khái quát hóa (generalized aggregator), huấn luyện theo mini-batch, và thuật toán lấy mẫu lân cận với kích thước cố định (fixed-size neighbor sampling) nhằm tăng tốc quá trình huấn luyện. Tuy nhiên, chiến lược lấy mẫu lân cận cố định không thể hoàn toàn tránh được vấn đề bùng nổ số lượng lân cận (neighborhood expansion problem). Ngoài ra, không có đảm bảo lý thuyết nào cho chất lượng của việc lấy mẫu.

## 3. Hiệu năng các kiến trúc GNN trong các nghiên cứu thực nghiệm về phân loại tài khoản mạng xã hội

Các nghiên cứu gần đây đã áp dụng GNN trên dữ liệu mạng xã hội thực tế (Twitter, Facebook…) để phân loại tài khoản thật - giả. Trong các phương pháp này, GCN, GAT và GraphSAGE là những mô hình GNN tiêu biểu được so sánh. Những mô hình này xem mỗi tài khoản người dùng là một đỉnh trong đồ thị và sử dụng các cạnh (quan hệ kết bạn, số lượt tương tác...) để học biểu diễn cho đỉnh, kết hợp với các đặc trưng của tài khoản (hồ sơ, nội dung bài viết…) nhằm dự đoán tài khoản đó là thật hay giả.

Nhiều benchmark đã được xây dựng từ dữ liệu thực tế để đánh giá hiệu năng mô hình. Chẳng hạn, TwiBot-20 (Twitter, 2020) có hơn 229.000 người dùng với khoảng (~5.800) bot. Kết quả tổng quan cho thấy các phương pháp dựa trên đồ thị (GNN) nhìn chung vượt trội hơn so với phương pháp truyền thống dùng đặc trưng rời rạc hoặc nội dung văn bản. Thực nghiệm trên TwiBot-20 cho thấy top 5 mô hình tốt nhất đều là GNN, với F1 trung bình cao hơn (~13.8%) so với mức trung bình của tất cả các phương pháp. Điều này khẳng định việc khai thác cấu trúc mạng xã hội bằng GNN giúp cải thiện đáng kể khả năng nhận diện tài khoản giả. Trong bài báo khoa học trên arXiv với mã số 2106.13092, có tiêu đề: *“BotRGCN: Twitter Bot Detection with Relational Graph Convolutional Networks”*được công bố bởi nhóm tác giả Shangbin Feng, Herun Wan, Ningnan Wang và Minnan Luo đến Xi’an Jiaotong University với kết quả thực nghiêm trên TwiBot-20 như sau:

A graph of different colored bars

Description automatically generated

**Hình 2.4:** Kết quả thực nghiệm bài báo mã số 2106.13092 trên arXiv

Kết quả cho thấy R-GCN đạt F1 cao nhất (~87%), nhỉnh hơn một chút so với GCN và GAT (~86%). Độ chính xác (accuracy) của các mô hình trong thử nghiệm này (~85%) và cao hơn đáng kể so với không sử dụng GNN (~81%). Tuy nhiên, hiệu năng tương đối của các kiến trúc trên còn phụ thuộc vào đặc điểm dữ liệu.

## **4. Ưu nhược điểm của các kiến trúc GNN**

### 4.1. Graph Convolutional Network - GCN

GCN áp dụng tích chập trên đồ thị bằng cách trung bình các hàng xóm của một đỉnh (theo ma trận Laplacian). Ưu điểm nổi bật của GCN là kiến trúc đơn giản, số lượng tham số ít nên dễ huấn luyện và ít nguy cơ quá khớp (overfitting) hơn trên dữ liệu ít. Thực tế cho thấy GCN có đủ cấu trúc phổ (spectrum) của đồ thị, phù hợp với các mạng đồng nhất cao (nơi kết nối giữa các nút cùng nhãn rất phổ biến – tính homophily cao).

Tuy nhiên, nhược điểm của GCN là tính chuyển tiếp (transductive) – mô hình cần được huấn luyện lại khi thêm nút mới, khó áp dụng trực tiếp cho tài khoản mới xuất hiện. GCN cũng đánh trọng số các láng giềng một cách đồng đều, không phân biệt được tầm quan trọng của từng hàng xóm. Điều này có thể làm giảm hiệu quả trên đồ thị mà thông tin từ các láng giềng không đồng nhất. Hơn nữa, phép tích chập dựa trên Laplacian khiến GCN bị ràng buộc bởi cấu trúc toàn cục (mọi láng giềng đóng góp ngang nhau theo bậc của nút). Trong trường hợp các tài khoản giả tinh vi ẩn mình giữa nhiều hàng xóm thật, GCN có thể bỏ lỡ tín hiệu đặc trưng do không biết tập trung vào một số kết nối quan trọng.

### 4.2. Graph Attention Network – GAT

GAT bổ sung cơ chế self-attention vào GNN, cho phép mô hình học trọng số khác nhau cho mỗi hàng xóm của một nút. Trong lớp GAT, mỗi nút *lắng nghe* thông tin từ các láng giềng với mức độ quan trọng khác nhau, được điều chỉnh qua hệ số attention học được. Ưu điểm rõ rệt của GAT là khả năng nhấn mạnh các kết nối quan trọng và giảm nhiễu từ các kết nối kém liên quan. Một ưu thế khác của GAT là linh hoạt với dữ liệu không đồng nhất với đồ thị nhiều kết nối khác nhãn (heterophily) hoặc có các loại quan hệ khác nhau, GAT vẫn có thể học trọng số phù hợp để phân biệt ảnh hưởng của từng loại hàng xóm. Nhờ khắc phục ràng buộc Laplacian, GAT tích hợp tốt hơn tương quan giữa các đặc trưng nút và đạt hiệu suất cao, đặc biệt trong việc phát hiện bot có hành vi khác biệt so với các hàng xóm.

Nhược điểm chính của GAT là chi phí tính toán cao hơn. Cơ chế attention đòi hỏi tính trọng số cho từng cạnhtrong mỗi bước lan truyền, khiến độ phức tạp tăng lên đáng kể khi một nút có nhiều hàng xóm. Trên đồ thị lớn (như mạng xã hội hàng triệu nút…) GAT có thể chậm hơn đáng kể so với GCN và GraphSAGE. Ngoài ra, GAT có nhiều tham số hơn (ma trận trọng số và vector attention), do đó mô hình phức tạp hơn và cần nhiều dữ liệu huấn luyện hơn để tránh quá khớp (overfitting). Trong một số thử nghiệm với dữ liệu ít, GAT ban đầu cho kết quả kém GCN, nhưng khi bổ sung thêm đặc trưng hoặc huấn luyện lâu hơn thì lại đạt hiệu quả cao hơn. Tóm lại, GAT thường đạt hiệu năng phân loại cao nhất nhờ trọng số linh hoạt nhưng đánh đổi bằng tốc độ và độ phức tạp triển khai.

### 4.3. Graph Sample and Aggregate - GraphSAGE

GraphSAGE là mô hình GNN quy nạp (inductive), học một hàm tập hợp (aggregate) để trích xuất đặc trưng từ mẫu lân cận của một nút. Lợi thế lớn nhất của GraphSAGE là khả năng tổng quát hóa cho các nút chưa từng thấy – rất hữu ích khi phát hiện tài khoản giả mới xuất hiện hoặc áp dụng mô hình qua các thời điểm khác nhau . GraphSAGE cũng cho phép huấn luyện theo mini-batch trên đồ thị lớn thay vì dùng toàn bộ đồ thị, GraphSAGE chỉ lấy một lượng hàng xóm ngẫu nhiên cho mỗi nút tại mỗi bước, giúp giảm yêu cầu bộ nhớ và tăng tốc độ trên mạng lớn. Tính năng neighbor sampling này là then chốtđể triển khai GNN trong môi trường thực tế, nơi cần cập nhật mô hình nhiều lần trong ngày.

Nhược điểm của GraphSAGE là việc lựa chọn chiến lược lấy mẫu lân cận và hàm tổng hợp có thể ảnh hưởng đến kết quả. Nếu mẫu lân cận quá nhỏ, mô hình có thể bỏ lỡ thông tin; nếu quá lớn, lại mất ưu thế tốc độ. So với GAT, GraphSAGE không học trọng số riêng cho từng cạnh, nên trong trường hợp một nút có nhiều hàng xóm và chỉ một vài trong số đó là quan trọng (ví dụ vài bạn bè thân thiết trong mạng xã hội…) GraphSAGE có thể không nhấn mạnh đúng mức các kết nối đó. Dù vậy, do không cần tính attention cho từng cạnh, GraphSAGE tiết kiệm tính toán hơn GAT trên đồ thị rất lớn và dễ triển khai song song. Tổng thể, GraphSAGE được đánh giá là giải pháp cân bằng giữa hiệu quả và chi phí, hiệu năng cao gần mức GAT mà huấn luyện nhanh hơn – phù hợp cho hệ thống quy mô lớn, động.

Các nghiên cứu thực nghiệm với dữ liệu mạng xã hội (Twitter, Facebook…) cho thấy GNN là phương pháp đầy hứa hẹn để phân loại tài khoản thật – giả. Các mô hình GCN, GraphSAGE, GAT đều đã được áp dụng thành công, giúp phát hiện nhiều tài khoản bot ẩn danh mà phương pháp truyền thống bỏ sót. Về hiệu năng, GAT thường nhỉnh hơn một chút về độ chính xác và F1-score nhờ cơ chế attention, GraphSAGE bám sát và có lợi thế về khả năng tổng quát hóa cho dữ liệu mới, còn GCNtuy đơn giản nhưng vẫn rất hiệu quả và đôi khi vượt trội trong hoàn cảnh nhất định. Việc lựa chọn mô hình tối ưuphụ thuộc vào đặc thù dữ liệu. Nếu đồ thị rất lớn và động, GraphSAGE là lựa chọn phù hợp. Nếu cần độ chính xác cao nhất và chấp nhận chi phí, GAT có thể nổi trội. Với dữ liệu hạn chế hoặc để làm đường cơ sở, GCN vẫn là giải pháp hiệu quả. Các kết quả mới nhất cũng gợi ý rằng kết hợp nhiều kỹ thuật (ví dụ R-GCN, mô hình lai GCN-GAT…) có thể nâng cao hơn nữa hiệu năng, khi đó GCN, GAT, GraphSAGE đóng vai trò nền tảng để xây dựng mô hình phức hợp.

# Chương 3: Cài đặt, huấn luyện mô hình, đánh giá hiệu quả

## 1. Môi trường và công cụ cài đặt

Trong quá trình nghiên cứu và triển khai các mô hình mạng nơ-ron đồ thị (Graph Neural Networks - GNN), việc thiết lập một môi trường cài đặt phù hợp là điều kiện tiên quyết nhằm đảm bảo tính ổn định, khả năng mở rộng và hiệu quả xử lý của hệ thống. Môi trường cài đặt trong đồ án này bao gồm ba yếu tố chính: ngôn ngữ và thư viện lập trình, nền tảng phần cứng – phần mềm, và cấu trúc triển khai các mô hình GNN.

### 1.1. Thư viện và framework sử dụng

Ngôn ngữ lập trình chính được sử dụng trong toàn bộ đồ án là Python 3.10, vốn là ngôn ngữ phổ biến trong lĩnh vực học máy và học sâu do cú pháp đơn giản, cộng đồng hỗ trợ rộng lớn và hệ sinh thái thư viện phong phú.

Các thư viện chính hỗ trợ triển khai mô hình bao gồm:

**- PyTorch:** Là một trong những framework học sâu mạnh mẽ và linh hoạt, PyTorch hỗ trợ định nghĩa mô hình dưới dạng dynamic computation graph, rất phù hợp với các mô hình phức tạp như GNN.

**- PyTorch Geometric (PyG):** Đây là thư viện mở rộng của PyTorch, chuyên biệt cho các bài toán xử lý dữ liệu đồ thị. PyG hỗ trợ nhiều lớp GNN như GCN (Graph Convolutional Network), GraphSAGE, GAT (Graph Attention Network), đồng thời cung cấp các hàm chuyển đổi dữ liệu, kỹ thuật sample dữ liệu hiệu quả và API thân thiện.

**- Scikit-learn:** Dùng để tính toán các độ đo đánh giá như độ chính xác (Accuracy), Precision, Recall, F1-score, đồng thời cung cấp công cụ để xây dựng ma trận nhầm lẫn (Confusion Matrix).

**- Matplotlib và Seaborn:** Hai thư viện trực quan hóa mạnh, được dùng để hiển thị kết quả huấn luyện, biểu đồ loss, accuracy theo epoch và ma trận nhầm lẫn.

**- NetworkX:** Dùng để biểu diễn, phân tích và trực quan hóa các cấu trúc đồ thị ở mức cơ bản.

Ngoài ra, một số thư viện phụ trợ như **NumPy**, **Pandas** cũng được sử dụng để xử lý dữ liệu đầu vào và trung gian.

### 1.2. Cấu hình phần cứng và phần mềm

Việc lựa chọn cấu hình phần cứng và phần mềm phù hợp đóng vai trò then chốt trong hiệu quả huấn luyện và thử nghiệm các mô hình mạng nơ-ron đồ thị (Graph Neural Networks - GNN). Toàn bộ quá trình triển khai trong đồ án được thực hiện trên máy tính có cấu hình như sau:

**Hệ điều hành:** Windows 11 Pro 64-bit

**CPU:** Intel(R) Core(TM) i7-4910MQ

**GPU**: NVIDIA Quadro K2100M

**RAM**: 32GB

Các thư viện học máy được cài đặt trong môi trường Python 3.10, được quản lý thông qua Conda với một môi trường ảo riêng biệt, giúp dễ dàng kiểm soát phiên bản và tránh xung đột giữa các thư viện phụ thuộc.

**1.3. Cấu trúc và cách tổ chức mô hình GNN**

Việc cài đặt mô hình GNN được thực hiện tuần tự với ba kiến trúc chính: GCN, GraphSAGE và GAT. Mỗi mô hình được thiết kế thành các lớp riêng biệt, bao gồm: lớp xử lý đầu vào, lớp GNN chính (convolution, aggregation), lớp phi tuyến (ReLU hoặc eLU), lớp dropout và lớp fully connected đầu ra.

Quy trình huấn luyện được tổ chức thành pipeline bao gồm các module:

Tiền xử lý dữ liệu

Tạo đối tượng đồ thị

Xây dựng mô hình

Huấn luyện và lưu mô hình

Đánh giá mô hình

Các mô hình được cài đặt với cùng các siêu tham số cơ bản (learning rate, số epoch, hàm mất mát...) nhằm đảm bảo tính công bằng khi so sánh kết quả thực nghiệm. Ngoài ra, quá trình huấn luyện được theo dõi bằng log epoch, loss, và accuracy, giúp kiểm tra hiện tượng overfitting hoặc underfitting.

## 2. Chuẩn bị dữ liệu

### 2.1. Mô tả bộ dữ liệu sử dụng

Trong nghiên cứu này, sử dụng bộ dữ liệu **Cresci-2015**, một trong những bộ dữ liệu uy tín và phổ biến trong lĩnh vực phát hiện tài khoản giả mạo (social bots) trên mạng xã hội. Bộ dữ liệu bao gồm hai phần chính:

**Tài khoản thật**: Là các tài khoản người dùng thật, được thu thập một cách ngẫu nhiên từ Twitter.

**Tài khoản giả**: Là tập hợp các tài khoản giả mạo hoặc bot được tạo ra có chủ đích để mô phỏng hành vi người dùng.

Các tập con E13 và INT trong bộ dữ liệu được kết hợp lại để hình thành một tập duy nhất gồm **2818 tài khoản**, trong đó mỗi tài khoản được gắn nhãn 0 (thật) hoặc 1 (giả). Ngoài ra, dữ liệu còn bao gồm thông tin về mối quan hệ giữa các tài khoản (friend/follower), cho phép xây dựng cấu trúc đồ thị xã hội để áp dụng các mô hình học sâu trên đồ thị (GNN).

### 2.2. Tiền xử lý dữ liệu

Quy trình tiền xử lý đóng vai trò quan trọng trong việc chuẩn bị dữ liệu cho huấn luyện mô hình, bao gồm ba giai đoạn chính: xử lý dữ liệu người dùng, xử lý dữ liệu kết nối (bạn bè) và chuyển đổi dữ liệu thành biểu diễn đồ thị.

#### 2.2.1. Làm sạch dữ liệu người dùng

Từ các tập tin CSV chứa thông tin người dùng (users.csv), tác giả chọn lọc một số cột đặc trưng có ý nghĩa cao trong phân tích, bao gồm: *statuses\_count, followers\_count, friends\_count, favourites\_count, listed\_count, lang, time\_zone, location*. Những cột này đại diện cho các đặc trưng định lượng và định tính mô tả hoạt động, mức độ phổ biến và thông tin ngôn ngữ của người dùng trên mạng xã hội.

Các bước xử lý bao gồm:

- Điền giá trị khuyết (NaN) bằng cách phù hợp với kiểu dữ liệu: giá trị số được điền bằng 0, chuỗi được mã hóa bằng LabelEncoder.

- Mã hóa tất cả đặc trưng dạng chuỗi thành dạng số nguyên để phục vụ mô hình học sâu.

- Loại bỏ các cột không cần thiết, chỉ giữ lại những đặc trưng đã chọn để huấn luyện.

- Kết quả là một bảng dữ liệu người dùng gồm **2818 dòng × 10 cột**, trong đó mỗi dòng là một người dùng duy nhất.

#### 2.2.2. Biểu diễn đồ thị từ dữ liệu bạn bè

Tập dữ liệu bạn bè (friends.csv) chứa các mối quan hệ giữa người dùng, được dùng để xây dựng các cạnh (edges) trong đồ thị. Mỗi dòng trong tập này là một cặp (source\_id, target\_id), thể hiện mối quan hệ "người này theo dõi người kia".

Các bước thực hiện:

- Ghép nối các file bạn bè từ hai tập con E13 và INT.

- Tạo chỉ mục ánh xạ từ ID người dùng sang chỉ số (index) trong mảng đặc trưng.

- Tạo danh sách các cạnh bằng cách ánh xạ từng cặp ID sang chỉ số.

- Loại bỏ các cạnh không hợp lệ như: cạnh tự nối (self-loop ban đầu) và cạnh trùng lặp.

- Tạo cạnh hai chiều để mô hình hóa mối quan hệ theo cả hai hướng.

Tổng cộng thu được khoảng **4560 cạnh hợp lệ**, thể hiện mối quan hệ giữa các người dùng trong mạng.

#### 2.2.3. Gán nhãn và chia tập huấn luyện, kiểm tra

Tập dữ liệu sau khi được biểu diễn dưới dạng đồ thị bao gồm:

**Tập đỉnh (nodes)**: tensor x ∈ ℝ^(2818×8) chứa đặc trưng người dùng.

**Tập cạnh (edges)**: tensor edge\_index ∈ ℤ^(2×7378) mô tả quan hệ.

**Nhãn (label)**: tensor y ∈ ℤ^2818 mô tả phân loại Real (0) hoặc Fake (1).

Sau khi chuẩn hóa và thêm self-loop vào đồ thị để đảm bảo tính ổn định khi lan truyền tín hiệu qua các lớp GCN, dữ liệu được đóng gói thành đối tượng Data của thư viện **PyTorch Geometric**.

Tiếp theo, dữ liệu được chia thành ba phần:

**Tập huấn luyện** (train): 70% tổng số đỉnh

**Tập xác thực** (validation): 15%

**Tập kiểm tra** (test): 15%

Việc chia dữ liệu sử dụng phương pháp stratified split để giữ nguyên phân bố nhãn giữa các tập. Các tập con này được biểu diễn bằng các mask dạng boolean: train\_mask, val\_mask, test\_mask và gán trực tiếp vào đối tượng graph.

### 2.3. Đặc trưng sử dụng cho các đỉnh (node features)

Các đặc trưng được trích xuất từ bảng người dùng gồm những thông tin định lượng dễ thu thập từ mạng xã hội. Cụ thể: