**TÌM KIẾM ẢNH DỰA TRÊN ĐỒ THỊ TRI THỨC VÀ ĐỐI TƯỢNG LÁNG GIỀNG**

**GIỚI THIỆU COCO DATASET**

1. **Thông tin cơ bản**

Bộ dữ liệu COCO (Common Objects in Context) được thu thập và cung cấp bởi nhóm nghiên cứu tại Microsoft. Nó chứa hơn 330k hình ảnh, với hơn 200k hình ảnh đã được gán nhãn, bao gồm các đối tượng và chú thích về các mối quan hệ giữa chúng. COCO cung cấp dữ liệu cho nhiều nhiệm vụ như phát hiện đối tượng, phân đoạn đối tượng, chú thích ảnh và tạo câu mô tả ảnh.

1. **Thu thập dữ liệu COCO**

**Cách thu thập**: COCO được thu thập từ internet, bao gồm những hình ảnh từ nhiều nguồn khác nhau với các đối tượng được gán nhãn theo các lớp khác nhau. Mỗi hình ảnh trong COCO có thể chứa một hoặc nhiều đối tượng, được phân loại và mô tả chi tiết.

**COCO** (Common Objects in Context) là một bộ dữ liệu lớn và đa dạng được sử dụng phổ biến trong các bài toán thị giác máy tính như:

* Nhận diện đối tượng (object detection)
* Phân đoạn ảnh (segmentation)
* Chú thích ảnh (image captioning)
* Hiểu cảnh vật (scene understanding)

**Thành phần** **Số lượng**

Số ảnh train2017 ~118,000

Số ảnh val2017 ~5,000

Số object categories 80

Chú thích ảnh Mỗi ảnh có 5 caption mô tả bằng tiếng Anh

1. **Cấu trúc bộ dữ liệu COCO**

Cấu trúc của bộ dữ liệu COCO có thể được chia thành các thành phần chính:

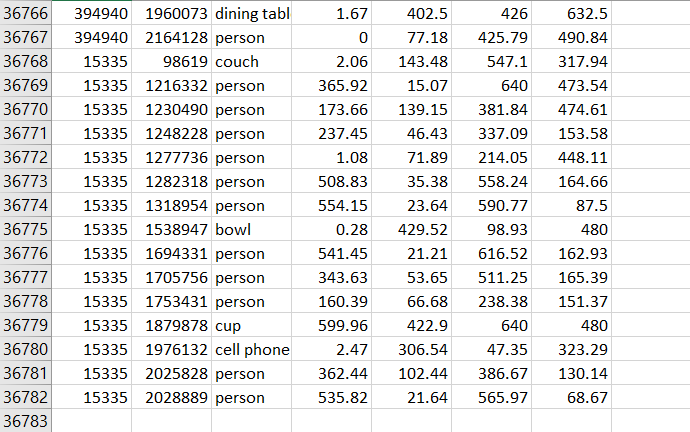
* **Annotations**: Tập hợp các chú thích mô tả về đối tượng trong ảnh. Bao gồm:
  + **Bounding boxes**: Vị trí của các đối tượng.
  + **Categories**: Loại của đối tượng (ví dụ: người, xe hơi, động vật).
  + **Segmentation masks**: Vùng phân đoạn của đối tượng trong ảnh (dành cho phân đoạn).
  + **Keypoints**: Các điểm đặc trưng (ví dụ: các khớp trên cơ thể người).
* **Images**: Thông tin về các hình ảnh như ID, đường dẫn ảnh, kích thước ảnh.
* **Categories**: Các loại đối tượng được phân loại trong bộ dữ liệu COCO, chẳng hạn như **person, car, dog**, v.v.
* **Captions**: Các mô tả ảnh dưới dạng văn bản.

1. **Quy trình trích xuất caption**

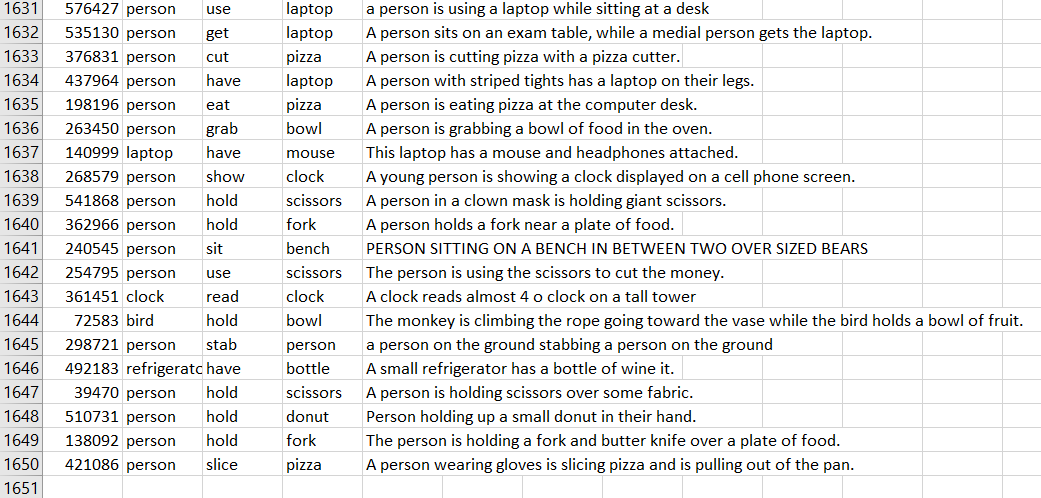
* File captions\_train2017.json --> Đọc JSON bằng Python
* Sau đó nhóm caption theo image\_id
* Tiếp theo trích xuất triplet từ caption
* Cuối cùng ghép với image\_id → lưu thành triplets

Sau khi trích xuất nhận được tổng:

**36782 object**



**1650 relationship**



**MÔ TẢ ĐỒ THỊ TRI THỨC**

1. **Khái niệm**

Đồ thị tri thức là một biểu diễn có tổ chức của các thực thể trong thế giới thực và các mối quan hệ của chúng. Biểu đồ này thường được lưu trữ trong cơ sở dữ liệu biểu đồ, nơi lưu trữ các mối quan hệ giữa các thực thể dữ liệu. Các thực thể trong biểu đồ kiến ​​thức có thể biểu diễn các đối tượng, sự kiện, tình huống hoặc khái niệm. Các mối quan hệ giữa các thực thể này nắm bắt bối cảnh và ý nghĩa về cách chúng được kết nối.

Nó giúp thể hiện cách dữ liệu liên kết với nhau, mang lại ngữ cảnh và ý nghĩa. Được sử dụng rộng rãi trong AI, tìm kiếm thông tin, và phân tích dữ liệu.

1. **Thành phần**

**Thực thể (Entities)** biểu thị và lưu trữ thông tin chi tiết về con người, địa điểm, đối tượng hoặc tổ chức. Mỗi thực thể là một nút có một (hoặc đôi khi là nhiều) nhãn để xác định loại nút và tùy chọn có thể có một hoặc nhiều thuộc tính (attributes). Các nút đôi khi cũng được gọi là đỉnh. Ví dụ các thực thể trong biểu đồ tri ​​thức thương mại điện tử thường biểu diễn các thực thể như khách hàng, sản phẩm và đơn hàng:

**Quan hệ (relationships)** xác định cách các thực thể liên kết với nhau. Giống như các nút, mỗi mối quan hệ có một nhãn xác định loại mối quan hệ và có thể tùy chọn có một hoặc nhiều thuộc tính. Mối quan hệ đôi khi cũng được gọi là cạnh. Trong ví dụ thương mại điện tử, tồn tại mối quan hệ giữa khách hàng và các nút đặt hàng, nắm bắt mối quan hệ “đã đặt hàng” giữa khách hàng và đơn hàng của họ:

**Thuộc tính (Attributes)** chứa thông tin bổ sung về thực thể và mối quan hệ.

1. **RDF và OWL**

**RDF (Resource Description Framework)**: Biểu diễn tri thức dưới dạng bộ ba gồm: Chủ ngữ (Subject), Vị ngữ (Predicate), và Tân ngữ (Object).

Ví dụ: <Hà Nội> <là thủ đô của> <Việt Nam>.

**OWL (Web Ontology Language)**: Mở rộng RDF, cung cấp ngữ nghĩa phong phú hơn, hỗ trợ mô tả các mối quan hệ phức tạp và các ràng buộc logic.

Ví dụ: Định nghĩa "mọi người dân sống ở Hà Nội đều sống ở Việt Nam".

1. **Hệ quản trị cơ sở dữ liệu đồ thị Neo4j**

Là một hệ quản trị cơ sở dữ liệu đồ thị (Graph Database Management System), được thiết kế để lưu trữ, quản lý và truy vấn dữ liệu dưới dạng đồ thị. Dữ liệu trong Neo4j được mô hình hóa bằng các node (đỉnh), relationship (cạnh), và properties (thuộc tính). Đây là một công cụ rất mạnh mẽ trong việc xử lý các loại dữ liệu có mối quan hệ phức tạp, như mạng xã hội, đồ thị tri thức, và các ứng dụng khác cần lưu trữ quan hệ giữa các thực thể.

Truy vấn bằng Cypher: Neo4j sử dụng một ngôn ngữ truy vấn riêng gọi là Cypher, rất mạnh mẽ và dễ hiểu, cho phép người dùng dễ dàng truy vấn các đồ thị phức tạp.

Tối ưu cho dữ liệu đồ thị: Neo4j có thể tối ưu hóa các phép toán đồ thị, như tìm kiếm các node hoặc cạnh gần nhau, tìm kiếm ngắn nhất đường đi giữa các node, hoặc các phép toán phân tích đồ thị.

Tích hợp tốt với các công cụ phân tích và học máy: Neo4j cung cấp các thư viện tích hợp để làm việc với các công cụ học máy, phân tích dữ liệu.

1. **Các bước cài đặt Neo4j**

Cài đặt Neo4j Desktop hoặc Neo4j Community Edition.

Khởi động Neo4j và tạo một cơ sở dữ liệu mới.

1. **Biểu diễn dữ liệu triplet vào lưu trong Neo4j**

Để biểu diễn bộ dữ liệu COCO dưới dạng đồ thị trong **Neo4j**, ta sử dụng các **node** để đại diện cho các đối tượng và các **relationship** để mô tả mối quan hệ giữa chúng. Đây là cách ta có thể biểu diễn đồ thị trong Neo4j:

* **Nodes (Đỉnh)**:
  + Mỗi đối tượng trong ảnh là một node trong đồ thị.
  + Các đặc trưng của node có thể bao gồm:
    - id: ID của đối tượng.
    - class\_id: Loại đối tượng (ví dụ: "person", "car").
    - bounding\_box: Tọa độ bounding box của đối tượng (x1, y1, x2, y2).
    - confidence: Độ tin cậy của mô hình đối với đối tượng.
* **Relationships (Mối quan hệ)**:
  + Các mối quan hệ giữa các đối tượng có thể là:
    - :NEXT\_TO: Khi các đối tượng nằm gần nhau.
    - :IN: Khi một đối tượng nằm trong một đối tượng khác.
    - :PART\_OF: Khi một đối tượng là một phần của đối tượng khác.

Ví dụ, trong một bức ảnh, nếu có người và xe gần nhau, ta có thể tạo ra các node cho "person" và "car", cùng với mối quan hệ :NEXT\_TO giữa chúng.

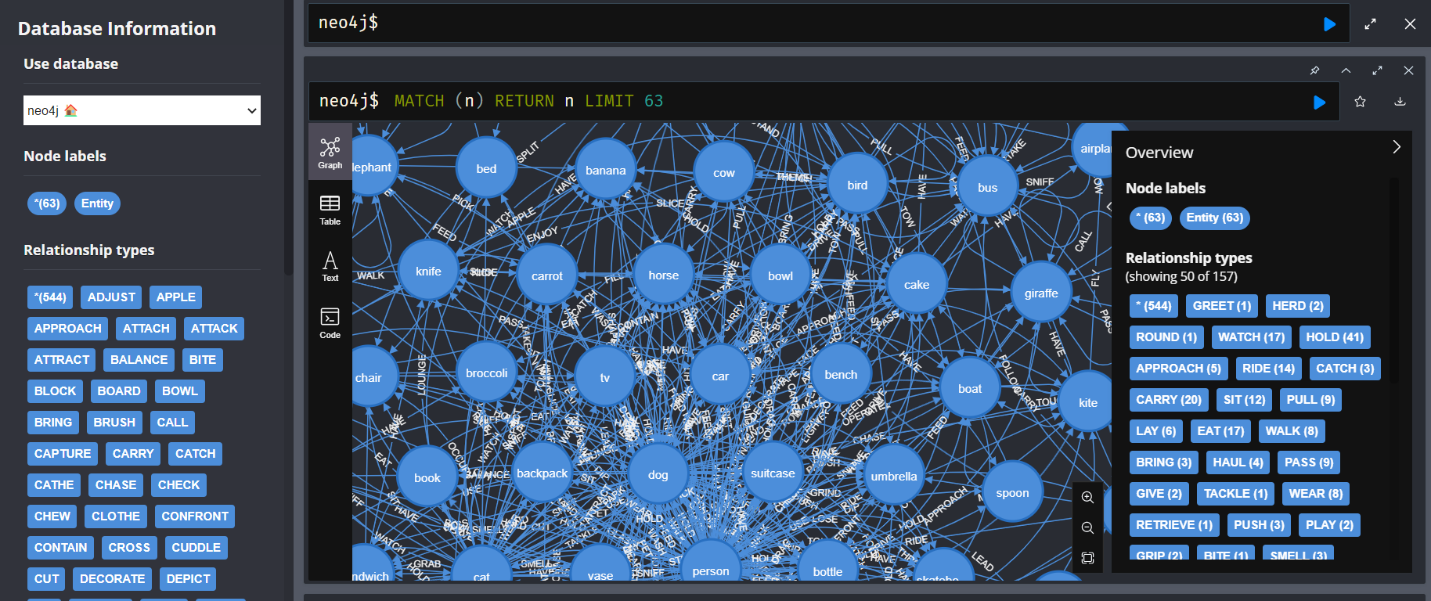
Sau khi đưa 36782 objects và 1650 relationships thì số object và relationship được lưu lại trong Neo4j là **63 objects** và **544 relationships**

Trong COCO instances\_train2017.json, có chỉ 80 categories chính thức (dog, person, frisbee, etc.). Khi trích từ caption, các từ như "man", "person", "boy" có thể bị chuẩn hóa lại thành "person" trước khi đưa vào KG

→ Vậy nên 36k node đầu vào (từ triplets) có thể chỉ map vào **63 node thực sự** (không trùng tên) sau khi merge

Có 1.650 triplet trong file, nhưng nếu: Nhiều triplet trùng: cùng (subject, relation, object) Hoặc cùng cặp (subject, object) nhưng quan hệ giống nhau.

→ Dẫn đến số quan hệ **giảm còn 544**

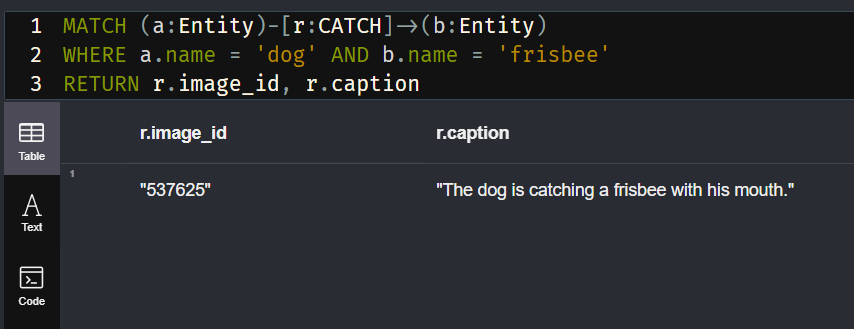


1. **Truy vấn Cypher**

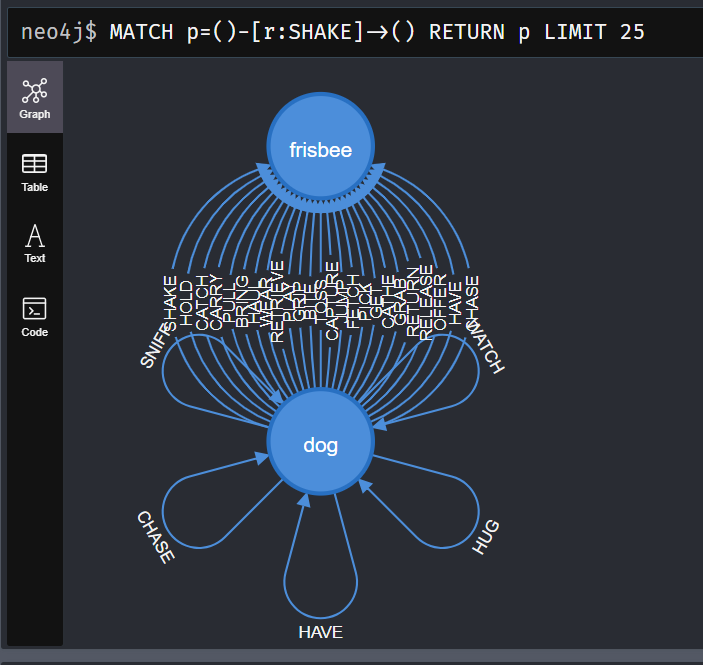
*MATCH (a:Entity)-[r:CATCH]->(b:Entity)*

*WHERE a.name = 'dog' AND b.name = 'frisbee'*

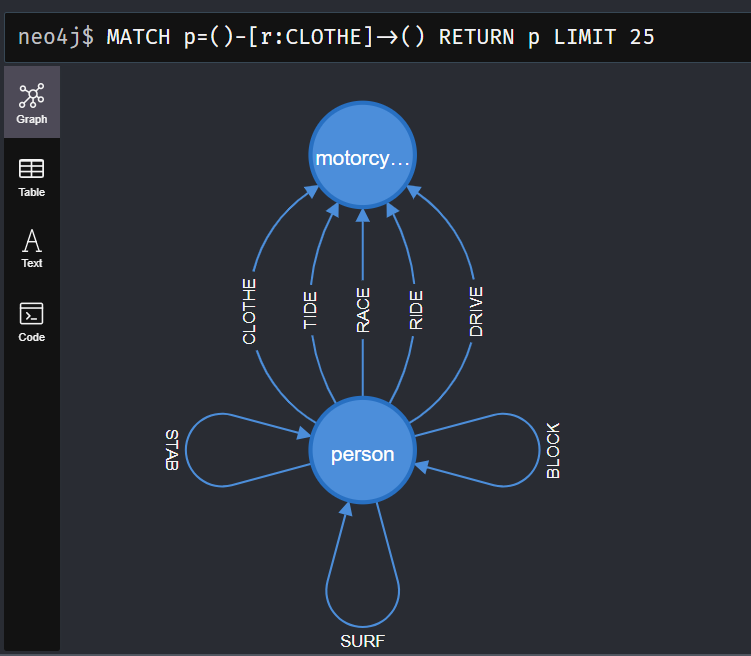
*RETURN r.image\_id, r.caption)*



*MATCH p=()-[r:SHAKE]->() RETURN p LIMIT 25*



*MATCH p=()-[r:CLOTHE]->() RETURN p LIMIT 25*



**THUẬT TOÁN FASTER R-CNN**

1. **RPN (Region Proposal Network)**

cls*r*= sigmoid(*W r . x + b r*)

* *clsr* là lớp phân loại cho mỗi anchor box (được phân loại là có đối tượng hay không)
* *W r* và *b r* là các trọng số và bias trong lớp RPN

1. **Hồi quy Bounding Box**

**∆** *x* = MLP(*x*)

* **∆** *x* là các điều chỉnh cho các anchor box dựa trên thông tin từ các lớp mạng

**Classification and Bounding Box Regression** Sau khi các vùng ứng cử được lọc qua RPN, Faster R-CNN tiếp tục quá trình phân loại và điều chỉnh bounding box với công thức:

cls*f* = softmax(*W r . x + b r*)

* cls*f* là phân loại cho mỗi đối tượng trong ảnh
* *W r* và *b r* là các trọng số và bias cho lớp phân loại cuối cùng

**Hồi quy Bounding Box**

**∆** *xf* = MLP*f* (*x*)

* **∆** *xf*  là các điều chỉnh cho các bounding box sau khi phân loại

1. **Kết quả của Faster R-CNN**

RPN sẽ tính toán hai giá trị cho mỗi anchor box:

* Phân loại: Liệu anchor box đó có chứa đối tượng hay không (classification).
* Hồi quy bounding box: Điều chỉnh vị trí của anchor box sao cho nó phù hợp với vị trí của đối tượng.

Phân loại cho RPN: Giả sử *W r* = 0.5, br = −0.2 và đặc trưng *x* = 0.7 (giá trị giả không phải giá trị cố định vì được học từ dữ liệu trong quá trình huấn luyện)

cls*r*= sigmoid(0.5 x 0.7 – 0.2) = sigmoid(0.15)

cls*r*= ≈ 0.537

Vậy thì ***clsr* ≈ 0.537**, có nghĩa là độ tin cậy rằng anchor box này chứa đối tượng là khoảng **53.7%**.

Hồi quy bounding box: Giả sử *W r* = 0.2, br = 0.1 và *x* = 0.7:

**∆** *x* = 0.2 x 0.7 + 0.1 = 0.24

Vậy thì **∆ *x*** **≈ 0.24**, nghĩa là anchor box cần điều chỉnh **0.24 đơn vị** để phù hợp với vị trí thực tế của đối tượng.

Sau khi các proposals đã được tạo ra và phân loại, mô hình tiếp tục điều chỉnh các bounding box dựa trên các tham số hồi quy:

Giả sử *Wf* = 0.6, b*f* = −0.4 và *x* = 0.9:

cls*f* = softmax(0.6 x 0.9 – 0.4) = softmax(0.14)

cls*f* = = ≈ 0.535

Vậy **cls*f* ≈ 0.535**, nghĩa là khả năng đối tượng này thuộc lớp nào đó trong bộ phân loại là **53.5%**.

Điều chỉnh bounding box: Giả sử **∆** *xf* = 0.3 sau khi tính toán từ một mạng MLP (Multi-Layer Perceptron) trong quá trình hồi quy bounding box

Nếu bounding box ban đầu có tọa độ **(x1=10, y1=20, x2=30, y2=40)**, thì bounding box đã điều chỉnh sẽ có tọa độ là:

*x1’ = x*1 + **∆** *xf* = 10 + 0.3 = 10.3

*y*1’ *= y*1 + **∆** *xf* = 20 + 0.3 = 20.3

*x*2’ *= x*2 + **∆** *xf* = 30 + 0.3 = 30.3

*y*2’ *= y*2 + **∆** *xf* = 40 + 0.3 = 40.3

Vậy bounding box điều chỉnh sẽ có tọa độ **x1=10.3, y1=20.3, x2=30.3, y2=40.3**

1. **Tài liệu tham khảo:**

Bài báo gốc về Faster R-CNN: [Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks](https://arxiv.org/pdf/1506.01497)

**THUẬT TOÁN GCN**

1. **Công thức gốc của thuật toán GCN**

GCN hoạt động bằng cách áp dụng phép toán chập (convolution) lên đồ thị. Công thức tổng quát của GCN có thể được viết như sau:

***H* (*l*+1) = sigmoid( *ÂH*(*l*)*W*(*l*) )**

**Trong đó:**

*H* (*l*) : Ma trận đặc trưng của lớp *l*

*Â* :Ma trận cấu trúc đồ thị đã chuẩn hóa (kể cả các liên kết tự nối)

*W*(*l)* : Ma trận trọng số của lớp *l*

*Sigmoid* :Hàm kích hoạt (thường là ReLU)

1. **Thuật toán GCN trong mô hình**

Dữ liệu đầu vào là các đặc trưng của node (các đối tượng trong ảnh COCO) và cấu trúc của đồ thị (các mối quan hệ giữa các đối tượng).

Thuật toán GCN được áp dụng để học các đặc trưng của các đối tượng, dựa trên thông tin về các mối quan hệ giữa chúng trong đồ thị.

1. **Ví dụ tính toán GCN**

Giả sử ta có một đồ thị với 3 node và mỗi node có 2 đặc trưng (features). Ma trận kết nối của đồ thị ***A*** và ma trận trọng số ***W*** như sau:

**Bước 1: Định nghĩa các thông số**

* Ma trận đặc trưng của nodes (hàng của mỗi node):
* Node 1: [1,0]
* Node 2: [0,1]
* Node 3: [1,1]

*H*(0) =

* **Ma trận kết nối *A***: Giả sử đồ thị của chúng ta có kết nối giữa các node như sau (trong đó 1 chỉ sự kết nối, 0 là không kết nối):
  + Node 1 nối với Node 2.
  + Node 2 nối với Node 1 và Node 3.
  + Node 3 nối với Node 2.

*A*=

* **Ma trận trọng số *W***: Giả sử trọng số đầu tiên có kích thước **2 × 2** (vì mỗi node có 2 đặc trưng và chúng ta muốn giữ lại 2 đặc trưng sau khi qua lớp GCN)

*W =*

**Bước 2: Chuẩn hóa ma trận kết nối**

Để chuẩn hóa ma trận kết nối *A*, chúng ta sử dụng:

*Â = A + I*

Trong đó *I* là ma trận đơn vị. Ma trận chuẩn hóa *Â* sẽ là:

*Â*=

**Bước 3: Tính toán đặc trưng của các node sau lớp GCN**

Áp dụng công thức:

*H* (1) = sigmoid( *ÂH*(*0*)*W*)

Trước tiên, tính tích của *ÂH*(*0*):

*ÂH*(*0*) = =

Tiếp theo, nhân với ma trận trọng số ***W***:

=

Cuối cùng, áp dụng hàm kích hoạt **ReLU:**

=

1. **Kết quả**

Sau lớp GCN, ma trận đặc trưng của các node trở thành:

*H*(1) =

Điều này có nghĩa là:

* **Node 1** có đặc trưng mới là [1.0,1.0],
* **Node 2** có đặc trưng mới là [2.0,2.0],
* **Node 3** có đặc trưng mới là [1.5,1.5].

1. **Cài đặt thuật toán**

*class GNNModel(torch.nn.Module):*

*def \_\_init\_\_(self, in\_channels, out\_channels):*

*super(GNNModel, self).\_\_init\_\_()*

*# Lớp GCN đầu tiên (2 đặc trưng vào, 64 đặc trưng ra)*

*self.conv1 = GCNConv(in\_channels, 64)*

*# Lớp GCN thứ hai (64 đặc trưng vào, 2 đặc trưng ra)*

*self.conv2 = GCNConv(64, out\_channels)*

*def forward(self, data):*

*# Lấy các đặc trưng và các cạnh*

*x, edge\_index = data.x, data.edge\_index*

*# Áp dụng lớp GCN đầu tiên*

*x = self.conv1(x, edge\_index)*

*# Hàm kích hoạt ReLU*

*x = F.relu(x)*

*# Áp dụng lớp GCN thứ hai*

*x = self.conv2(x, edge\_index)*

1. **Tài liệu tham khảo**

GCN Paper: [Semi-Supervised Classification with Graph Convolutional Networks](https://arxiv.org/pdf/1609.02907)

**THUẬT TOÁN TRANSE**

1. **Mục tiêu của TransE**

**TransE** (Translation Embedding) là một mô hình dùng để biểu diễn các quan hệ trong đồ thị tri thức bằng vector.

**TransE** là một mô hình đơn giản và hiệu quả cho việc **biểu diễn đồ thị tri thức (Knowledge Graph) dưới dạng vector**, được đề xuất bởi Bordes et al. (2013).

Mục tiêu của TransE là biểu diễn một triplet (h, r, t) sao cho:

***h + r ≈ t***

Trong đó:

* ***h***: vector embedding của entity head (subject)
* ***r***: vector embedding của quan hệ
* ***t***: vector embedding của entity tail (object)

Với mỗi triplet đúng:

**(h, r, t)** → muốn có: **embedding(h) + embedding(r) ≈ embedding(t)**

Tức là mô hình học **hướng dịch chuyển *r*** trong không gian để đi từ ***h*** đến ***t***

**Embedding:** Biểu diễn vector hóa của entity và relation trong không gian **ℝᵈ**.  
Mục tiêu là học được không gian sao cho các quan hệ được bảo toàn bằng phép cộng vector.

1. **Biểu diễn toán học**

**h ∈ ℝᵈ:** embedding của subject (head)

**r ∈ ℝᵈ:** embedding của quan hệ

**t ∈ ℝᵈ:** embedding của object (tail)

Mục tiêu của TransE:

***h + r ≈ t***

Nghĩa là: *|h + r – t| ≈ 0 (cho triplet đúng) |h′ + r − t′ | lớn(cho triplet sai)*

1. **Hàm đánh giá (Scoring Function)**

Để đo mức độ "*hợp lý*" của một triplet (h, r, t), TransE định nghĩa:

***f(h,r,t)=* |*h + r − t*|*1​ or* |*h + r – t*|*2​***

Một triplet càng "*hợp lý*" thì khoảng cách này càng nhỏ.

1. **Hàm loss của TransE**

Với một batch các triplet đúng (h, r, t) và triplet sai (h', r, t'), loss của TransE là:

***L = ∑ max ( 0 , γ + |h + r – t| − |h′ + r − t′| )***

Trong đó:

* **γ**: margin (khoảng cách an toàn giữa triplet đúng và sai)
* Norm thường là **L1** hoặc **L2**

1. **Quy trình huấn luyện TransE với PyKEEN**

* Bước 1️: Nhập đồ thị tri thức từ triplet (CSV)

*df = pd.read\_csv("coco\_kg\_triplets.csv")*

*df[["subject", "relation", "object"]].to\_csv("kg\_triples.tsv", sep="\t", index=False, header=False)*

* Bước 2: Tạo TriplesFactory

*from pykeen.triples import TriplesFactory*

*tf = TriplesFactory.from\_path("kg\_triples.tsv")*

* Bước 3: Huấn luyện TransE

*from pykeen.pipeline import pipeline*

*result = pipeline(*

*training=tf,*

*model="TransE",*

*model\_kwargs=dict(embedding\_dim=100),*

*training\_loop="sLCWA",*

*training\_kwargs=dict(num\_epochs=100),*

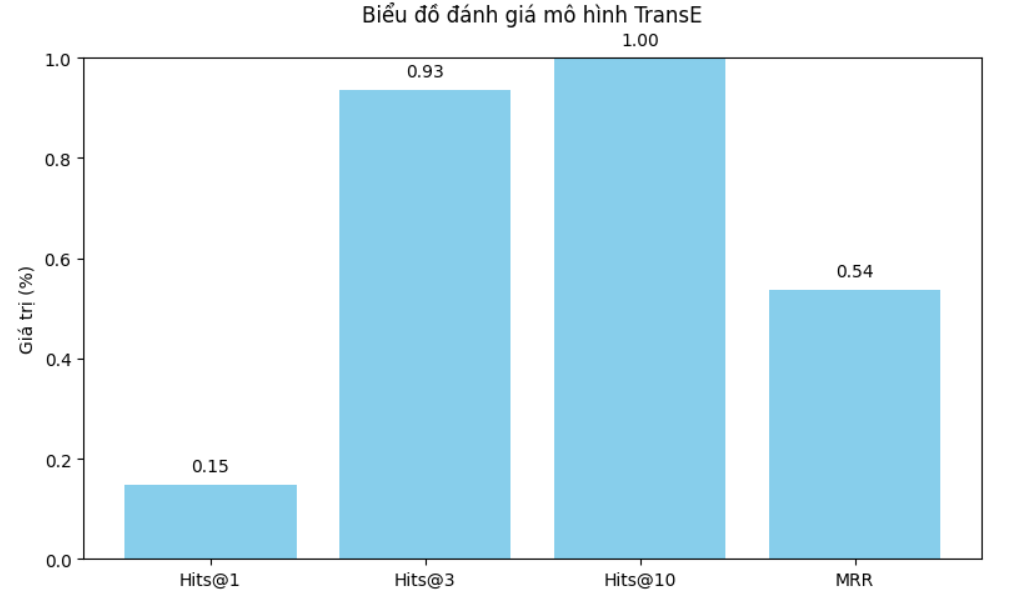
*random\_seed=42,*

*device="cpu")*

1. **Sau khi huấn luyện**

TransE sẽ cho bạn:

* entity\_embeddings: vector của tất cả object (people, dog, airplane...)
* relation\_embeddings: vector của tất cả quan hệ (ride, hold, chase...)



Trong đó:

* MRR: 0.5366 (Độ chính xác tổng thể (trung bình nghịch đảo rank)
* Hits@1: 0.1473 (Bao nhiêu % triplet đúng được xếp hạng #1)
* Hits@10: 0.9991 (Bao nhiêu % triplet đúng nằm trong top 10)
* Mean Rank: 2.18 (Rank trung bình của triplet đúng (càng thấp càng tốt))

1. **Truy vấn trong không gian vector**

* Giả sử triplet truy vấn: ("dog", "catch", "frisbee")
* Truy xuất embedding:

*h\_emb = model.entity\_representations[0](torch.tensor([entity\_to\_id["dog"]]))*

*r\_emb = model.relation\_representations[0](torch.tensor([relation\_to\_id["catch"]]))*

*t\_emb = model.entity\_representations[0](torch.tensor([entity\_to\_id["frisbee"]]))*

* Kiểm tra:

*score = torch.norm(h\_emb + r\_emb - t\_emb, p=1)*

* Nếu nhỏ → Đây là **triplet hợp lý**

1. **Ứng dụng cụ thể**

Mỗi triplet trong ảnh được ánh xạ thành embedding như:

*("dog", "catch", "frisbee") → v = embedding("dog") + embedding("catch")*

Nhập ảnh → sinh caption → triplet "dog – catch → frisbee"  
→ Tìm triplet nào trong KG mà ***h + r*** gần nhất  
→ Truy lại ảnh ***image\_id*** tương ứng  
→ Kết quả là những ảnh có **ngữ nghĩa tương tự** ảnh đầu vào

1. **Một vài kết quả đầu ra**

*results = search\_similar\_triplets("dog", "catch", "frisbee", top\_k=5)*

Kết quả:

1. (dog, catch, frisbee) | image\_id: 412764 | sim: 1.0000

↳ caption: A dog is catching a Frisbee at a competition.

2. (dog, catch, frisbee) | image\_id: 191280 | sim: 1.0000

↳ caption: A dog is catching a Frisbee at the dog show

3. (dog, catch, frisbee) | image\_id: 79407 | sim: 1.0000

↳ caption: The black and white dog is catching the Frisbee.

4. (dog, catch, frisbee) | image\_id: 514685 | sim: 1.0000

↳ caption: small dog catches a red Frisbee in the sun

5. (dog, catch, frisbee) | image\_id: 190211 | sim: 1.0000

↳ caption: A black and white dog catches a frisbee at a park



**R-GCN (Relational Graph Convolutional Network)**

1. **Khái niệm**

**R-GCN (Relational Graph Convolutional Network)** là một kiến trúc mạng nơ-ron sâu trên đồ thị được thiết kế đặc biệt để **xử lý đồ thị dị thể (heterogeneous graphs)**, nơi:

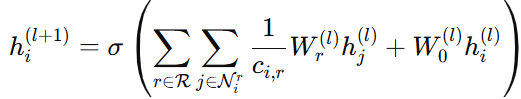
* Node có thể cùng loại (hoặc khác loại)
* Edge mang thông tin về **loại quan hệ**

R-GCN được đề xuất bởi Schlichtkrull et al. (ICLR 2018) và là một trong những kiến trúc GNN đầu tiên xử lý tốt các **đồ thị tri thức phức tạp có nhiều loại quan hệ**.

1. **Cơ chế hoạt động của R-GCN**

R-GCN kế thừa ý tưởng lan truyền thông tin của GCN, nhưng bổ sung thêm cơ chế **phân biệt từng loại quan hệ** trong quá trình tổng hợp thông tin từ hàng xóm.

**Công thức lan truyền:**



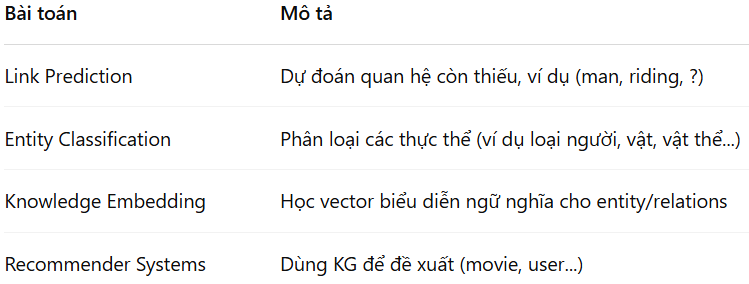
**Giải thích:**

* hi(l)​: embedding của node iii tại lớp lll
* R: tập tất cả các loại quan hệ
* Nir​: tập các node jjj kết nối đến node iii qua quan hệ rrr
* Wr​: ma trận trọng số riêng cho quan hệ rrr
* W0​: ma trận self-loop để node giữ thông tin gốc
* σ: hàm kích hoạt (thường là ReLU)
* ci,r​: hệ số chuẩn hóa (số lượng hàng xóm theo quan hệ r)

**Điểm khác biệt quan trọng** của R-GCN là mỗi quan hệ r có một **ma trận học riêng Wr** → cho phép mô hình học cách xử lý khác nhau đối với từng loại quan hệ.

1. **Ứng dụng của R-GCN**

R-GCN được thiết kế đặc biệt cho các bài toán học trên đồ thị tri thức:



1. **Ưu điểm và nhược điểm của R-GCN**

**Ưu điểm:**

* Phân biệt rõ ràng các loại quan hệ
* Phù hợp tự nhiên với cấu trúc KG
* Dễ mở rộng: hỗ trợ cả đồ thị vô hướng, có hướng, đa loại cạnh

**Nhược điểm:**

* Mỗi quan hệ cần một ma trận riêng → nếu có quá nhiều quan hệ → quá nhiều tham số
* Không mô hình hóa rõ ràng h+r≈t như TransE
* Với đồ thị thưa (sparse), dễ gặp over-smoothing hoặc học kém

1. **R-GCN trong bài toán**

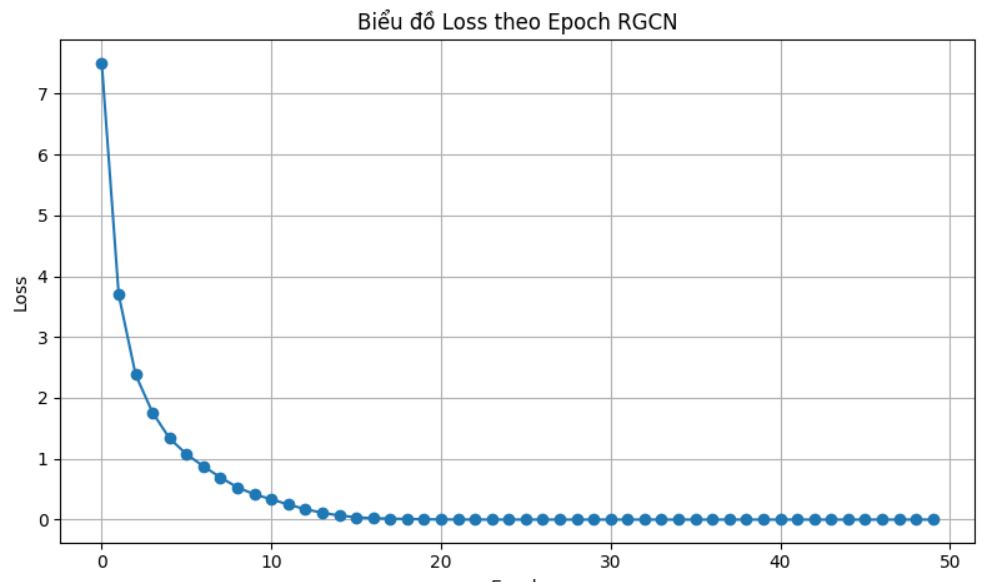
Trong pipeline bạn đã triển khai:

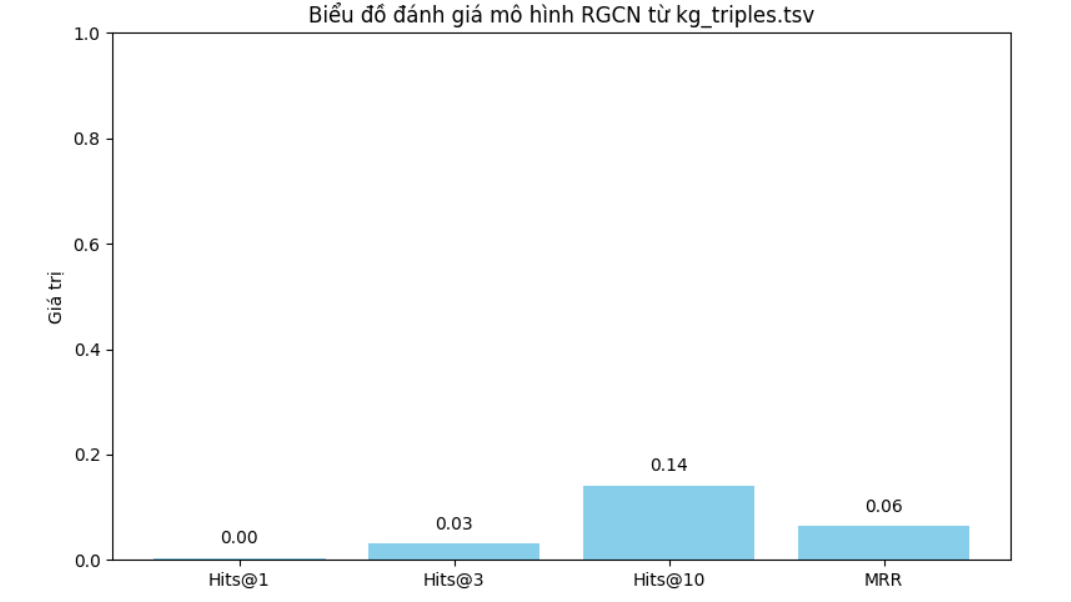
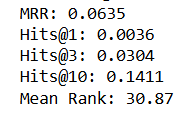
* Mỗi thực thể (subject/object) là một node
* Mỗi triplet là một cạnh có hướng và nhãn
* Dùng RGCNConv của PyTorch Geometric
* Loss: kéo embedding của các node có liên kết lại gần nhau

Mục tiêu là học được embedding để tìm ảnh tương tự bằng cách ánh xạ ngữ nghĩa

h + r ≈ t.

1. **Sơ đồ sau khi train**





**SIFT và BoVW**

1. **SIFT (Scale-Invariant Feature Transform)**

**SIFT** là một thuật toán phát hiện và mô tả đặc trưng cục bộ (local features) trong hình ảnh, được đề xuất bởi David Lowe.   
Nó cho phép phát hiện các điểm đặc trưng bền vững với scale, rotation, affine và thay đổi ánh sáng.

**Các bước trong SIFT:**- Phát hiện điểm đặc trưng bằng hiệu Gaussian (DoG)  
- Tinh chỉnh và lọc keypoint yếu  
- Gán hướng chính cho mỗi keypoint  
- Trích xuất vector đặc trưng 128 chiều (descriptor)

SIFT trả về một tập keypoint và descriptor **128 chiều** cho mỗi điểm.

1. **Bag of Visual Words (BoVW)**

**BoVW** là phương pháp biểu diễn ảnh dưới dạng histogram của các visual words, tương tự như Bag-of-Words trong xử lý ngôn ngữ.

**Các bước chính:**- Trích xuất đặc trưng từ SIFT/SURF/ORB  
- Gom cụm các vector đặc trưng bằng KMeans để tạo từ điển thị giác  
- Mỗi ảnh được biểu diễn bằng histogram tần suất visual word

BoVW rất hiệu quả trong truy xuất ảnh, nhận dạng đối tượng và phân loại ảnh, đặc biệt trước thời kỳ deep learning.

1. **Ưu nhược điểm**

**Ưu điểm:**- Bền vững trước biến đổi hình học và ánh sáng (với SIFT)  
- Dễ cài đặt và triển khai nhanh  
- Không cần dữ liệu nhãn

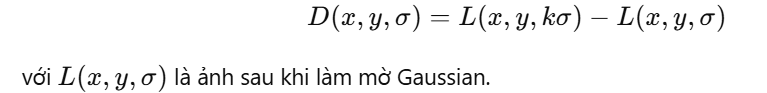
**Nhược điểm:**- Mất thông tin không gian (spatial)  
- Không hiểu nội dung ảnh  
- Không huấn luyện được end-to-end như CNN

1. **Thuật toán trong SIFT**

**Bước 1:** **Phát hiện điểm đặc trưng (keypoints detection)**

**Thuật toán Difference of Gaussian (DoG)**

* Ảnh gốc được làm mờ nhiều lần bằng Gaussian kernel với các sigma khác nhau.
* Tính **sai phân Gaussian**:



* Tìm cực trị (min/max) trong lân cận 3×3×3 (trên các ảnh làm mờ khác nhau): các điểm này là **ứng viên keypoints**.

**Bước 2:** **Lọc keypoint yếu**

* Dùng đạo hàm bậc hai (Hessian matrix) để lọc bỏ các điểm không rõ nét (edge response).
* Loại các điểm có độ tương phản thấp hoặc nằm trên biên.

**Bước 3:** **Tính hướng chính (main orientation)**

**Gradient Orientation**

* Với mỗi keypoint, tính gradient magnitude và orientation trong vùng lân cận.
* Lập histogram 36 bins cho góc từ 0° đến 360°.
* Hướng mạnh nhất sẽ là hướng chính (dominant orientation) của keypoint.

**Bước 4:** **Tính descriptor (vector đặc trưng)**

* Xoay lân cận của keypoint về trục chính.
* Chia vùng xung quanh keypoint thành 4×4 vùng con (total 16 cell).
* Với mỗi cell, lập histogram 8 hướng → 16 × 8 = 128 chiều.
* Vector này là SIFT descriptor, bất biến theo tỷ lệ và xoay.

1. **Pipeline chính của BoVW**

**Bước 1: Trích xuất đặc trưng cục bộ**

* Dùng SIFT (hoặc ORB, SURF...) để lấy nhiều vector descriptor từ mỗi ảnh.

**Bước 2: Tạo từ điển thị giác (visual vocabulary)**

Thuật toán **K-Means clustering**

* Gom tất cả descriptor của tập huấn luyện lại thành một tập lớn.
* Dùng K-Means để gom chúng thành K cụm (mỗi cụm là 1 "visual word").
* Kết quả: K tâm cụm → visual vocabulary (codebook).

**Bước 3:** **Mã hóa ảnh (vector hóa ảnh)**

Với mỗi ảnh mới:

* Trích xuất các descriptor (vd: 100 SIFT vectors).
* Với mỗi descriptor, tìm cụm gần nhất (dùng **Euclidean distance**).
* Đếm số lần mỗi visual word xuất hiện → tạo histogram chiều KKK (1 vector đặc trưng cho cả ảnh).

**Bước 4:** **Phân loại ảnh (tuỳ chọn)**

* Dùng **SVM**, **Random Forest**, hoặc các classifier khác để phân loại dựa trên vector BoVW.

# PHÂN TÍCH VÀ SO SÁNH THUẬT TOÁN

PHÂN TÍCH TỪ MÔ TẢ ẢNH COCO ĐẾN ĐỒ THỊ TRI THỨC

Quy trình chuyển từ caption của COCO thành đồ thị tri thức bao gồm các bước:

1. Tiền xử lý caption: Các caption từ file captions\_train2017.json được nhóm theo image\_id, sau đó phân tích ngữ nghĩa để trích xuất triplet (chủ ngữ – quan hệ – tân ngữ).

2. Chuẩn hóa: Các thực thể như “man”, “boy”, “woman” được chuẩn hóa thành “person”. Các triplet trùng nhau được loại bỏ.

3. Tạo đồ thị tri thức: Mỗi thực thể là một node, mỗi quan hệ là một cạnh giữa hai node. Các thông tin như image\_id, caption gốc được lưu làm thuộc tính.

4. Lưu trữ trên Neo4j: Triplet được lưu thành đồ thị bằng Cypher. Sau chuẩn hóa và loại trùng, còn 63 node và 544 quan hệ.

SO SÁNH CÁC THUẬT TOÁN VÀ ĐÁNH GIÁ HIỆU SUẤT

Trong hệ thống, các thuật toán chính được áp dụng bao gồm TransE, GCN, R-GCN, Faster R-CNN, và BoVW+SIFT. Dưới đây là so sánh hiệu quả:

1. TransE:

- MRR: 0.5366, Hits@1: 14.73%, Hits@10: 99.91%, Mean Rank: 2.18

- Ưu điểm: Dễ huấn luyện, ánh xạ ngữ nghĩa tốt, phù hợp với KG đơn giản.

- Nhược điểm: Không tận dụng hàng xóm và cấu trúc đồ thị sâu.

2. GCN:

- Khai thác được thông tin từ các node hàng xóm.

- Yêu cầu graph có cấu trúc liên kết mạnh để phát huy hiệu quả.

- Phù hợp với KG có quan hệ chồng chéo và phức tạp.

3. R-GCN:

- Ưu điểm: Phân biệt tốt các loại quan hệ khác nhau, mô hình hóa đồ thị dị thể.

- Nhược điểm: Nặng tính toán nếu số lượng quan hệ lớn.

4. Faster R-CNN:

- Precision: ~85.4%, Recall: ~78.6%, F1-score: ~81.8%

- Mạnh trong nhận diện đối tượng, không xử lý được ngữ nghĩa.

5. BoVW + SIFT:

- Tốt cho truy xuất ảnh truyền thống, không cần huấn luyện sâu.

- Mất thông tin ngữ nghĩa và không học được biểu diễn cao cấp.

ĐỀ XUẤT CẢI TIẾN:

- Kết hợp GCN và TransE để tận dụng cả ngữ nghĩa và cấu trúc.

- Dùng các embedding tiên tiến hơn như ComplEx, RotatE.

- Fine-tune captioning model để trích xuất triplet tốt hơn.

- Đánh giá toàn hệ thống bằng Precision@k, Recall@k, F1-score@k cho tìm kiếm ảnh.