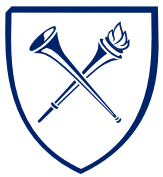
# Scene Graph based Visual Retrieval

## Paper Reading

Hejie Cui

Department of Computer Science, Emory University

December 7, 2021



EMORY UNIVERSITY

1. Image Retrieval
2. Graph Matching
3. Reference

1 Image Retrieval
(a) Paper 1: Image-to-Image Retrieval by Learning Similarity between Scene Graphs [YKJ+21]
(b) Comments and Other Related Works

2 Graph Matching

3 Reference

1 Image Retrieval
(a) Paper 1: Image-to-Image Retrieval by Learning Similarity between Scene Graphs [YKJ+21]
(b) Comments and Other Related Works

2 Graph Matching

3 Reference

# Image Retrieval with Scene Graph Similarity (IRSGS)

**Background:**

- Image Retrieval: finding similar images to a query image from a database (visual search engines)
- Existing work focus on image with a clear representative object
- CNN based method focus on superficial and low-level features

**Challenges:**

- no public available labeled data for complex image-to-image retrieval
- need a similarity measure to reflect semantics of images, i.e., the context and relationship of entities in images

# Image Retrieval with Scene Graph Similarity (IRSGS)

**Contributions:**

- Utilizes the similarity between scene graphs computed from a graph neural network to retrieve semantically similar images
- Collect more that 10,000 human annotations over 1,700 image triplets to evaluate and publish
- Train with the surrogate captions relevance obtained from a pre-trained language model (costly to collect enough gt for training)


Figure 1: Image retrieval examples from ResNet and IRSGS. ResNet retrieves images with superficial similarity, e.g., grayscale or vertical lines, while IRSGS successfully returns images with correct context, such as playing tennis or skateboarding.

# Two-Step Retrieval using Visual Features

- A common practice in information retrieval: retrieve roughly relevant items first and then re-rank the retrieved ones
- Step 1 : For a query image, first retrieve $K = 100$ images that are closest to the query in a ResNet-152 feature representation space formed by the 2048-dimension activation vector of the last hidden layer. The distance is measured in cosine similarity
- Step 2: Perform IRSGS for re-ranking (focus of the paper)

# Overview of the Proposed Method

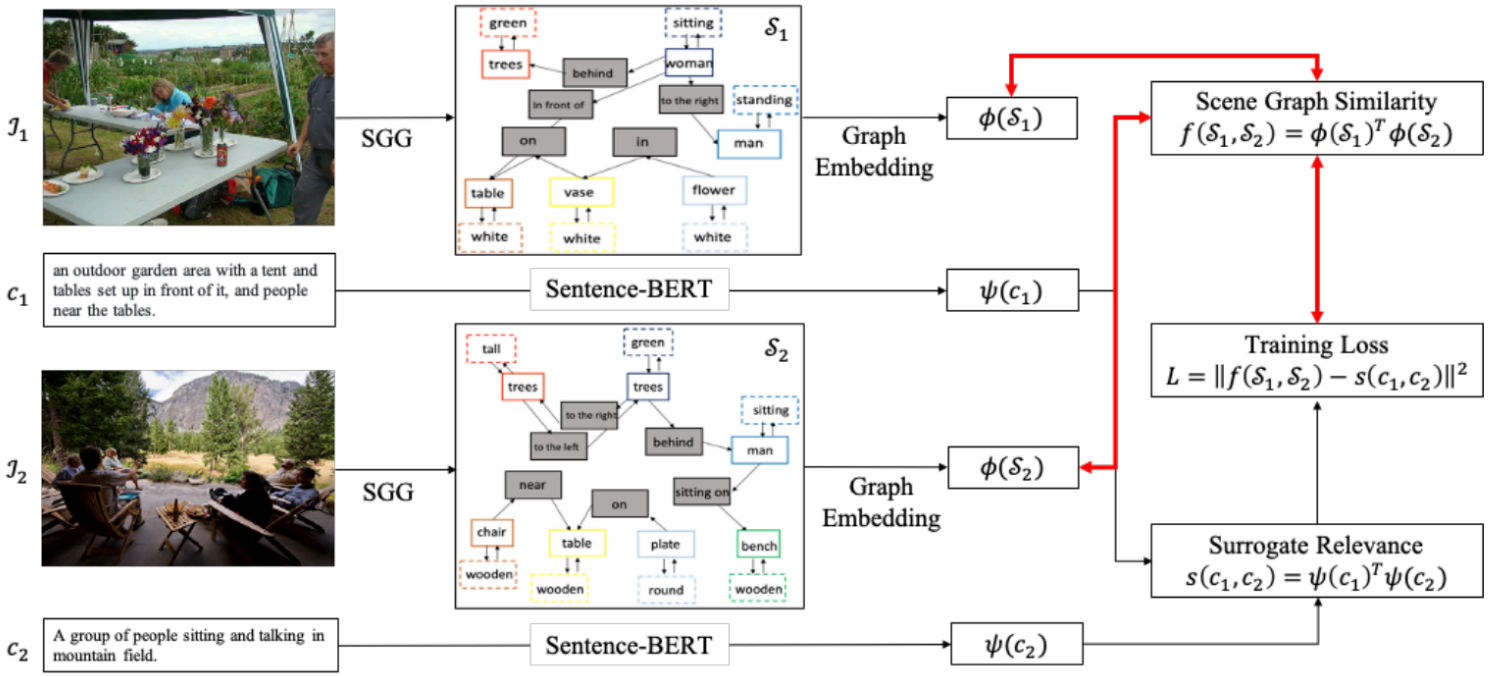



Figure 2: An overview of IRSGS. Images $\mathcal{I}_1, \mathcal{I}_2$ are converted into vector representations $\phi(\mathcal{S}_1), \phi(\mathcal{S}_2)$ through scene graph generation (SGG) and graph embedding. The graph embedding function is learned to minimize mean squared error to surrogate relevance, i.e., the similarity between captions. The bold red bidirectional arrows indicate trainable parts. For retrieval, the learned scene graph similarity function is used to rank relevant images.

# Scene Graphs and Their Generation

- A scene graph $\mathcal{S} = \{\mathcal{O}, \mathcal{A}, \mathcal{R}\}$ of an image $\mathcal{I}$ is defined as a set of objects $\mathcal{O}$, attributes of objects $\mathcal{A}$, and relations on pairs of objects $\mathcal{R}$.
- Treat all objects, attributes, and relations as nodes, and the associations among them as undirected edges.
- **Scene Graph Generation**
  - objects in image are detected by Faster R-CNN, and the name and attributes of the objects are predicted based on ResNet-101 features from the detected bounding boxes
  - for each pair of the detected objects, relations are predicted based on the frequency prior knowledge from the GQA dataset

# Retrieval via Scene Graph Similarity

- Given a query image $\mathcal{I}_q$, an image retrieval system ranks candidate images $\{\mathcal{I}_i\}_{i=1}^N$ according to the similarity to the query image sim($\mathcal{I}_i$, $\mathcal{I}_q$).
- Cast image retrieval into graph retrieval problem

$$sim(\mathcal{I}_i, \mathcal{I}_j) = f(\mathcal{S}_i, \mathcal{S}_j)$$

- The scene graph similarity is given as:

$$f(\mathcal{S}_1, \mathcal{S}_2) = \phi(\mathcal{S}_1)^\top \phi(\mathcal{S}_2),$$

where $\phi$ is constructed by computing the forward pass of GNNs (two versions: IRSGS-GCN and IRSGS-GIN) to obtain node representations and then apply average pooling.

# Learning to Predict Surrogate Relevance

**Surrogate relevance measure**: let $c_i$ and $c_j$ are the captions of image $\mathcal{I}_i$ and $\mathcal{I}_j$. Apply Sentence-BERT to obtain representation vectors $\psi(c_i)$ and $\psi(c_j)$. The surrogate relevance measure is given by inner product:

$$s(c_i, c_j) = \psi(c_i)^\top \psi(c_j)$$

**Loss function**: train the scene graph similarity $f$ by directly minimizing mean square error from the surrogate relevance measure, formulating the learning as a regression problem:

$$L_{ij} = \|f(\mathcal{S}_i, \mathcal{S}_j) - s(c_i, c_j)\|^2$$

# Dataset

**VG-COCO:**

- each image has a scene graph annotation provided by Visual Genome and five captions provided by MS-COCO
- 35,017 training images and 13,203 test images
- query set: randomly select 1,000 images among the test set

**Flickr30k:**

- five captions are provided per an image
- self generated scene graphs
- 30,000 training, 1000 validation and 1,000 testing (query set)

# Human Annotation Collection

**Annotated Triplets Selection Criterion**: visually close yet semantically different image triplets:

- Top 100 candidate images based on the cosine similarity in ResNet152 representation to the query image
- The surrogate relevance of a query-candidate image pair in a triplet should be larger than the other (diff $> 0.1$)

**Human Agreement Score** Given a triplet, s1 (or s2 ) is the number of human annotators who chose the 1st (or the 2nd) candidate image is more semantically similar to the query, s3 be ... who answered that all three images are identical, and s4 be ... who marked the candidates as irrelevant

$$\frac{s_i + 0.5s_3}{s_1 + s_2 + s_3 + s_4},$$

where $i = 1$ if the algorithm determines that the first image is semantically closer and $i = 2$ otherwise

# Evaluation

The relevance of images ranked by the algorithm is evaluated with two metrics:

- normalized discounted cumulative gain (nDCG) with the surrogate relevance as gain
- the agreement between a retrieval algorithm and decision of human annotators

# Quantitative Results

Baselines:

- Generated Caption
- ResNet-152 Features and ResNet Finetune
- Object Count (OC)
- Graph Matching Networks (GMN)

| Method | Data | nDCG | | | | | | Human Agreement |
|---|---|---|---|---|---|---|---|---|
| | | 5 | 10 | 20 | 30 | 40 | 50 | |
| Inter Human | - | - | - | - | - | - | - | $0.730 \pm 0.05$ |
| Caption SBERT | Cap(HA) | 1 | 1 | 1 | 1 | 1 | 1 | 0.700 |
| Random | - | 0.136 | 0.138 | 0.143 | 0.147 | 0.149 | 0.152 | $0.472 \pm 0.01$ |
| Gen. Cap. SBERT | Cap(Gen) | 0.609 | 0.628 | 0.657 | 0.681 | 0.703 | 0.726 | 0.473 |
| ResNet | I | 0.687 | 0.689 | 0.691 | 0.692 | 0.693 | 0.693 | 0.494 |
| ResNet-FT | I | 0.642 | 0.656 | 0.682 | 0.703 | 0.724 | 0.745 | 0.478 |
| Object Count | I+SG | 0.736 | 0.749 | 0.770 | 0.788 | 0.804 | 0.819 | 0.587 |
| GMN | I+SG | 0.721 | 0.735 | 0.755 | 0.771 | 0.786 | 0.801 | 0.535 |
| IRSGS-GIN | I+SG | 0.751 | 0.768 | 0.790 | 0.808 | 0.824 | 0.839 | 0.576 |
| IRSGS-GCN | I+SG | **0.784** | **0.795** | **0.814** | **0.829** | **0.844** | **0.856** | **0.602** |

Table 1: Image retrieval results on VG-COCO with human-annotated scene graphs. Data column indicates which data modalities are used. Cap(HA): human-annotated captions. Cap(Gen): machine-generated captions. I: image. SG: scene graphs.

# Qualitative Results

- ResNet neglects the semantics and focuses on the superficial visual characteristics of images
- OC only accounts for the presence of objects
- IRSGS ontains images with similar objects with similar relations


Figure 3: Four most similar images retrieved by six algorithms. OC: Object Count, GIN: IRSGS-GIN, GCN: IRSGS-GCN. The visual genome ids for the query images are 2323522 and 2316427.

# Ablation Study

From the IRSGS-GCN framework:

- Ignore Attributes
- Randomize Relations

| Method | nDCG | | | | Human Agreement |
|---|---|---|---|---|---|
| | 5 | 10 | 20 | 40 | |
| IRSGS-GCN | 0.771 | 0.784 | 0.805 | 0.836 | 0.611 |
| No Attribute | 0.767 | 0.782 | 0.803 | 0.834 | 0.606 |
| Random Relation | 0.764 | 0.777 | 0.797 | 0.828 | 0.604 |
| Object Count | 0.730 | 0.743 | 0.761 | 0.794 | 0.581 |

Table 4: Scene graph component ablation experiment results on VG-COCO. Machine-generated scene graphs are used.

1. Both cases are better than OC that uses only object information.
2. Relations are important for capturing the human perception of semantic similarity.

1 Image Retrieval
(a) Paper 1: Image-to-Image Retrieval by Learning Similarity between Scene Graphs [YKJ+21]
(b) Comments and Other Related Works

2 Graph Matching

3 Reference

**Comments**

- use only naive dot product for graph embedding similarity
- the model to produce graph level embedding is simply GCN and GIN
- based on only the semantics contained in the two single images, didn't include any external commonsense knowledge

**Techniques from text-to-image retrieval**

- first perform alignment between text and image, then construct a alignment graph and apply GNN [DZML21]
- attention mechanism for filtration
- utilize a global node [PB15]
- utilize an inference graph based on Bayesian Network to complete the entity attribute graph [XZW+19]

1 Image Retrieval

2 Graph Matching
(a) Paper 2: Graph Matching Networks for Learning the Similarity of Graph Structured Objects [LGD+19]
(b) Comments and Other Related Works

3 Reference

1 Image Retrieval

2 Graph Matching
(a) Paper 2: Graph Matching Networks for Learning the Similarity of Graph Structured Objects [LGD+19]
(b) Comments and Other Related Works

3 Reference

# Similarity Learning for Graph Structured Objects

**Background:**

- How GNNs can be trained to produce embedding of graphs in vector spaces that enables efficient similarity reasoning
- Applications: similarity based retrieval in graph database (Fig.1)
- A successful model should
  - exploit graph structure
  - reason about the similarity of graphs from learned semantics as well

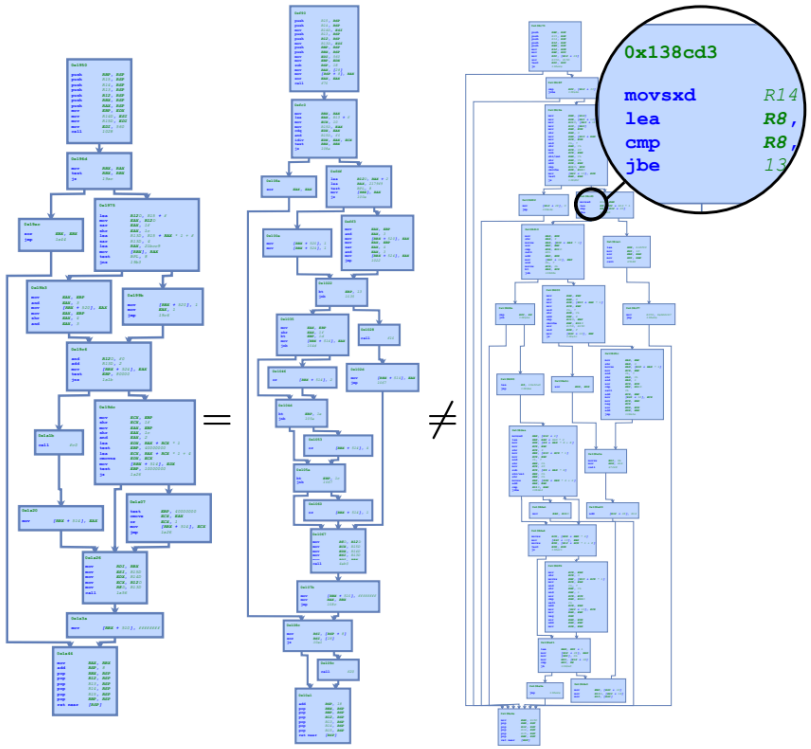



*Figure 1.* The binary function similarity learning problem. Checking whether two graphs are similar requires reasoning about both the structure as well as the semantics of the graphs. Here the left two control flow graphs correspond to the same function compiled with different compilers (and therefore similar), while the graph on the right corresponds to a different function.

# Related Concepts for Graph Similarity Learning

**Graph kernels:**

- Kernels on graphs designed to capture graph similarity, hand-designed and motivated by graph theory
- Typically formulated as first computing the feature vectors for each graph (kernel embedding), and then take inner product to compute kernel value
- Popular Graph Kernels
  - kernels that measure the similarity between walks or paths
  - kernels based on limited-sized sub-structures
  - kernels based on sub-tree structures

# Related Concepts for Graph Similarity Learning (Cont.)

**Distance metric learning:**

- Early work: data already lies in a vector space and learn a metric matrix to properly measure the distance in this space to group similar examples together and dissimilar ones apart
- Recently have been combined with applications such as face verification
- Our focus: similarity metric learning for graphs

**Siamese networks:**

- A family of NN for visual similarity learning
- Two networks with shared parameters applied to two input images independently to compute representation and fuse them with a small network

# Contributions

- Demonstrate how GNNs can be used to produce graph embeddings for similarity learning
- Propose the new Graph Matching Networks (GMNs) that computes similarity through cross-graph attention-based matching
- The proposed models outperform structure agnostic models and established hand-engineered baselines

# Graph Embedding Models v.s. Graph Matching Networks

Graph embedding models

- embed each graph independently into a vector and then all the similarity computation happens in the vector space

Graph matching networks

- instead of computing graph representations independently for each graph, the GMNs compute a similarity score through a cross-graph attention mechanism to associate nodes across graphs and identify differences

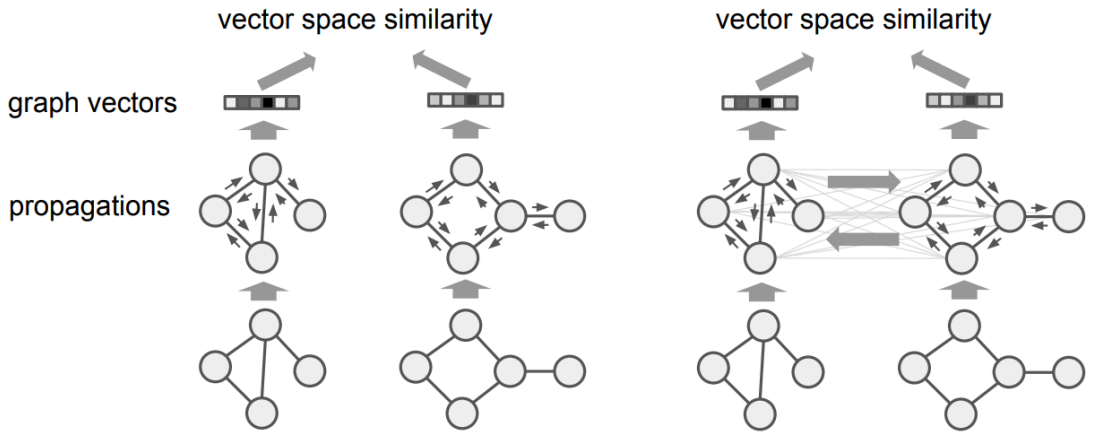



*Figure 2.* Illustration of the graph embedding (left) and matching models (right).

# Graph Embedding Models

**Encoder**

$$\mathbf{h}_i^{(0)} = \text{MLP}_{\text{node}}\left(\mathbf{x}_i\right), \quad \forall i \in V$$
$$\mathbf{e}_{ij} = \text{MLP}_{\text{edge}}\left(\mathbf{x}_{ij}\right), \quad \forall (i,j) \in E$$

**Propagation Layers** map a set of node representations $\left\{\mathbf{h}_i^{(t)}\right\}_{i \in V}$ to new node representations $\left\{\mathbf{h}_i^{(t+1)}\right\}_{i \in V}$

$$\mathbf{m}_{j \to i} = f_{\text{message}}\left(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}, \mathbf{e}_{ij}\right)$$
$$\mathbf{h}_i^{(t+1)} = f_{\text{node}}\left(\mathbf{h}_i^{(t)}, \sum_{j:(j,i)\in E} \mathbf{m}_{j \to i}\right)$$

**Aggregator** take the set of node representations $\left\{\mathbf{h}_i^{(T)}\right\}$ of $T$ rounds of propagations as input, and compute a graph level representation $\mathbf{h}_G = f_G\left(\left\{\mathbf{h}_i^{(T)}\right\}\right)$

$$\mathbf{h}_G = \text{MLP}_G\left(\sum_{i \in V} \sigma\left(\text{MLP}_{\text{gate}}\left(\mathbf{h}_i^{(T)}\right)\right) \odot \text{MLP}\left(\mathbf{h}_i^{(T)}\right)\right)$$

# Graph Matching Networks

**Propagation Layers** The main difference lays in the propagation layer: utilize a cross-graph matching vector to measure how well a node in one graph can be matched to one or more nodes in the other (independent $\rightarrow$ jointly)

$$
\begin{aligned}
\mathbf{m}_{j\rightarrow i} &= f_{\text{message}}\left(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}, \mathbf{e}_{ij}\right), \forall (i,j) \in E_1 \cup E_2 \\
\boldsymbol{\mu}_{j\rightarrow i} &= f_{\text{match}}\left(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}\right) \\
&\qquad \forall i \in V_1, j \in V_2, \text{ or } i \in V_2, j \in V_1 \\
\mathbf{h}_i^{(t+1)} &= f_{\text{node}}\left(\mathbf{h}_i^{(t)}, \sum_j \mathbf{m}_{j\rightarrow i}, \sum_{j'} \boldsymbol{\mu}_{j'\rightarrow i}\right) \\
\mathbf{h}_{G_1} &= f_G\left(\left\{\mathbf{h}_i^{(T)}\right\}_{i\in V_1}\right) \\
\mathbf{h}_{G_2} &= f_G\left(\left\{\mathbf{h}_i^{(T)}\right\}_{i\in V_2}\right) \\
s &= f_s\left(\mathbf{h}_{G_1}, \mathbf{h}_{G_2}\right)
\end{aligned}
$$

# Graph Matching Networks (Cont.)

$f_s$ is a standard vector space similarity between $\mathbf{h}_{G_1}$ and $\mathbf{h}_{G_2}$. $f_{\text{match}}$ is a function that communicates cross graph information, which the paper propose to use an attention based module:

$$a_{j\to i} = \frac{\exp\left(s_h\left(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}\right)\right)}{\sum_{j'}\exp\left(s_h\left(\mathbf{h}_i^{(t)}, \mathbf{h}_{j'}^{(t)}\right)\right)} \tag{1}$$

$$\boldsymbol{\mu}_{j\to i} = a_{j\to i}\left(\mathbf{h}_i^{(t)} - \mathbf{h}_j^{(t)}\right)$$

and therefore,

$$\sum_j \boldsymbol{\mu}_{j\to i} = \sum_j a_{j\to i}\left(\mathbf{h}_i^{(t)} - \mathbf{h}_j^{(t)}\right) = \mathbf{h}_i^{(t)} - \sum_j a_{j\to i}\mathbf{h}_j^{(t)} \tag{2}$$

$s_h$ is a vector space similarity metric like Euclidean or cosine similarity. $a_{j\to i}$ are the attention weights. $\sum_j \boldsymbol{\mu}_{j\to i}$ intuitively measures the difference between $\mathbf{h}_i^{(t)}$ and its closest neighbor in the other graph.

# Learning

Margin-based pairwise loss:

- train on pairs $(G_1, G_2)$

$$L_{\text{pair}} = \mathbb{E}_{(G_1, G_2, t)} \left[\max\left\{0, \gamma - t\left(1 - d\left(G_1, G_2\right)\right)\right\}\right]$$

Margin-based triplet loss:

- train on triplets $(G_1, G_2, G_3)$

$$L_{\text{triplet}} = \mathbb{E}_{(G_1, G_2, G_3)} \left[\max\left\{0, d\left(G_1, G_2\right) - d\left(G_1, G_3\right) + \gamma\right\}\right]$$

# Exp1: Synthetic Graph Edit Distances Learning

**Problem Background**

- graph edit distance between graphs $G_1$ and $G_2$ is defined as the min num of edit operations needed to transform $G_1$ to $G_2$
- NP-hard in general, therefore approximations have to be used

**Training Setup**

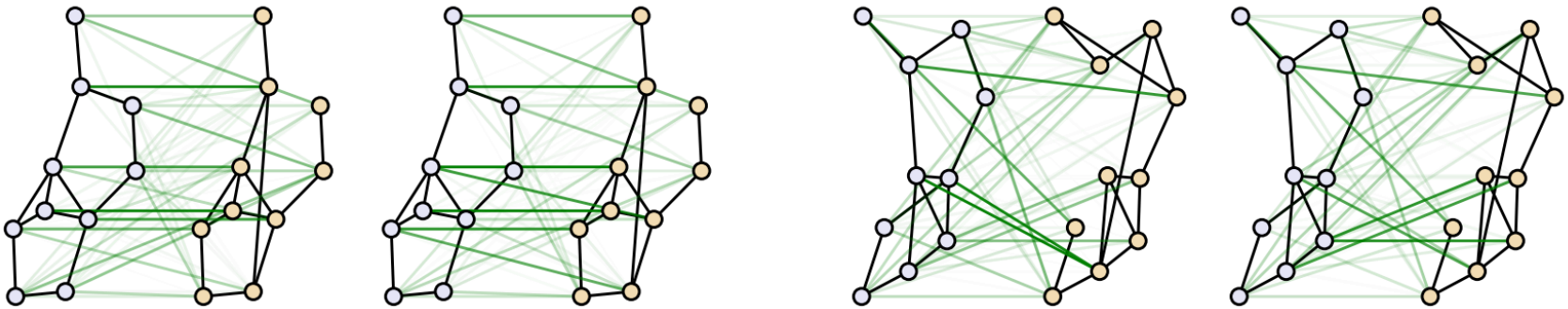
- generated training data by sampling random binomial graphs $G_1$ with $n$ nodes and edge probability $p$
- create positive example $G_2$ by randomly substituting $k_p$ edges from $G_1$ with new edges, and negative example $G_3$ by substituting $k_n$ edges from $G_1$, where $k_p < k_n$
- a model needs to predict a higher similarity score for positive pair $(G_1, G_2)$ than negative pair $(G_1, G_3)$

# Exp1: Synthetic Graph Edit Distances Learning (Cont.)

## Results

| Graph Distribution | WL kernel | GNN | GMN |
|---|---|---|---|
| $n = 20, p = 0.2$ | 80.8 / 83.2 | 88.8 / 94.0 | **95.0 / 95.6** |
| $n = 20, p = 0.5$ | 74.5 / 78.0 | 92.1 / 93.4 | **96.6 / 98.0** |
| $n = 50, p = 0.2$ | 93.9 / **97.8** | 95.9 / 97.2 | **97.4** / 97.6 |
| $n = 50, p = 0.5$ | 82.3 / 89.0 | 88.5 / 91.0 | **93.8 / 92.6** |

*Table 1.* Comparing the graph embedding (GNN) and matching (GMN) models trained on graphs from different distributions with the baseline, measuring pair AUC / triplet accuracy ($\times 100$).



graph edit distance = 1 graph edit distance = 2

*Figure 3.* Visualization of cross-graph attention for GMNs after 5 propagation layers. In each pair of graphs the left figure shows the attention from left graph to the right, the right figure shows the opposite.

# Exp2: Control Flow Graph based Binary Function Similarity Search

**Problem Background**

- Control Flow Graph (CFG) contains all the information in a binary function in a structured format
- CFGs for the same function have high similarity and low similarity otherwise

**Training Setup and Baseline**

- vulnerability search: given a binary known to have some vulnerabilities as query, search through a library to find similar binaries that may have the same vulnerabilities
- baselines: Google's open source function similarity search tool (hand-engineered graph hashing process)

# Exp2: CFG based Binary Function Similarity Search (Cont.)

## Results

- metrics: pair AUC and triplet accuracy

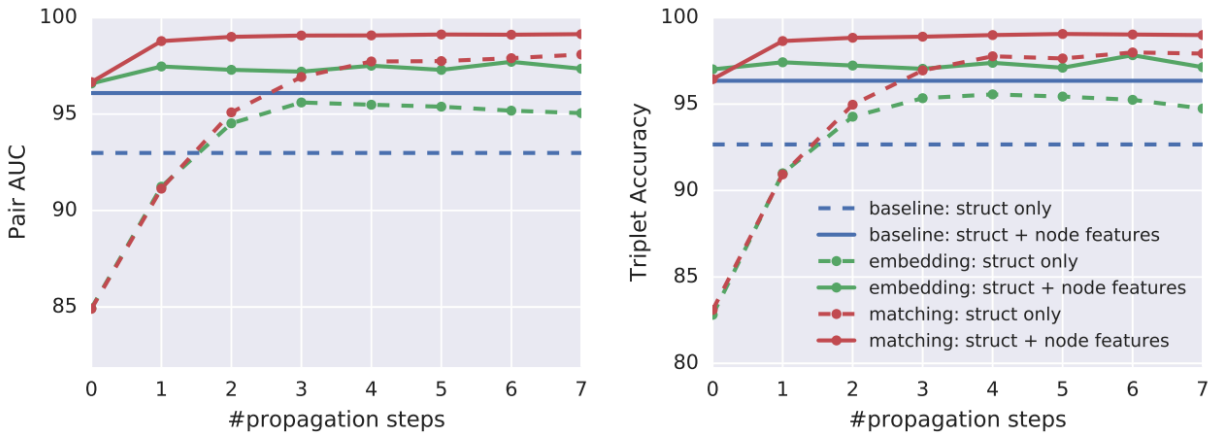



*Figure 4.* Performance (×100) of different models on the binary function similarity search task.

# Exp3: More Baselines and Ablation Studies

## More Baselines

- Graph Convolutional Network (GCN) model
- Siamese versions of the GNN/GCN embedding models

## Extra Dataset

- COIL-DEL mesh graph dataset
- mesh retrieval

| Model | Pair AUC | Triplet Acc |
|---|---|---|
| Baseline | 96.09 | 96.35 |
| GCN | 96.67 | 96.57 |
| Siamese-GCN | 97.54 | 97.51 |
| GNN | 97.71 | 97.83 |
| Siamese-GNN | 97.76 | 97.58 |
| GMN | **99.28** | **99.18** |

Function Similarity Search

| Model | Pair AUC | Triplet Acc |
|---|---|---|
| GCN | 94.80 | 94.95 |
| Siamese-GCN | 95.90 | 96.10 |
| GNN | 98.58 | 98.70 |
| Siamese-GNN | 98.76 | 98.55 |
| GMN | **98.97** | **98.80** |

COIL-DEL

*Table 2.* More results on the function similarity search task and the extra COIL-DEL dataset.

# Conclusion

**Strength:** The added power for the graph matching models comes from the fact that they are not independently mapping each graph to an embedding, but rather doing comparisons at all levels across the pair of graphs

**Shortness:**

- each cross-graph matching step requires the computation of the full attention matrices, which requires at least $O(|V1||V2|)$ time, this may be expensive for large graphs
- the matching models operate on pairs, and cannot directly be used for indexing and searching through large graph databases

1 Image Retrieval

2 Graph Matching
(a) Paper 2: Graph Matching Networks for Learning the Similarity of Graph Structured Objects [LGD+19]
(b) Comments and Other Related Works

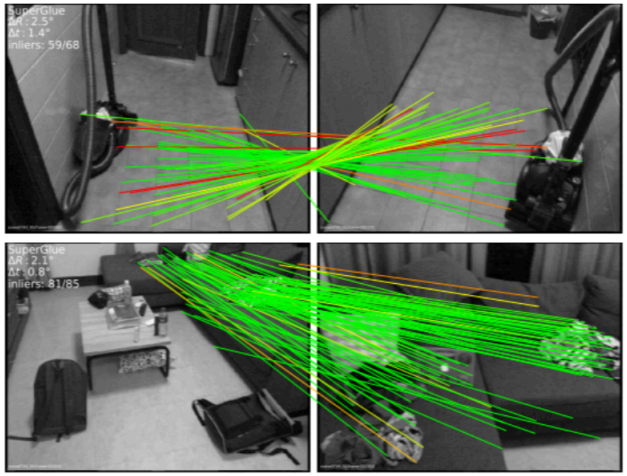
3 Reference

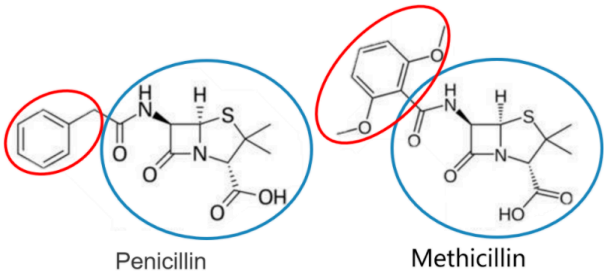
# Comments

- cross graph attention is helpful for graph matching since it provides more fine-grained matching
- more efficient cross-graph attention mechanisms are needed for graph matching, e.g. sparse connection
- for scene graph matching in image retrieval, edge (relation) attributes and frequency can be further considered
- the understanding of GNN's discriminative power (isomorphism test): as powerful as the Weisfeiler-Lehman algorithm in terms of distinguishing graphs

# Other Related Works

- current graph matching is mainly studied in computer vision application such as image correspondence, image key point matching
- combinatorial optimization, e.g. combinatorial embedding networks [WYY19]
- deep parametric feature hierarchies [ZS18]





1 Image Retrieval

2 Graph Matching

3 Reference

[DZML21] Haiwen Diao, Ying Zhang, Lin Ma, and Huchuan Lu.
Similarity reasoning and filtration for image-text matching.
In *AAAI*, 2021.

[LGD+19] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli.
Graph matching networks for learning the similarity of graph structured objects.
In *ICML*, 2019.

[PB15] Nikita Prabhu and R. Venkatesh Babu.
Attribute-graph: A graph based approach to image ranking.
In *ICCV*, 2015.

[WYY19] Runzhong Wang, Junchi Yan, and Xiaokang Yang.
Learning combinatorial embedding networks for deep graph matching.
In *ICCV*, 2019.

[XZW+19] Peixi Xiong, Huayi Zhan, Xin Wang, Baivab Sinha, and Ying Wu.
Visual query answering by entity-attribute graph matching and reasoning.
In *CVPR*, 2019.

[YKJ+21] Sangwoong Yoon, Woo-Young Kang, Sungwook Jeon, SeongEun Lee, Changjin Han, Jonghun Park, and Eun-Sol Kim.
Image-to-image retrieval by learning similarity between scene graphs.
In *AAAI*, 2021.

[ZS18] Andrei Zanfir and Cristian Sminchisescu.
Deep learning of graph matching.
In *CVPR*, 2018.

*Thanks!*