Interpretable Neural Subgraph Matching for Graph Retrieval (Appendix)

A Additional related work

Our work is closely related to the graph matching problems in computer vision, combinatorial approaches in graph and subgraph matching, and graph neural networks. We briefly review them in the following sections.

A.1 Graph matching in computer vision

The closely related problem of deep graph matching has been heavily investigated by the computer vision community. However, their existing neural matching models generally require explicit supervision of node level ground truth alignment. This is used for computing a variety of losses, such as displacement loss (Zanfir and Sminchisescu 2018), permutation loss (Wang, Yan, and Yang 2019; Tan et al. 2021; Zhao, Tu, and Xu 2021; Fey et al. 2020) or Hungarian attention loss (Yu et al. 2019). Such domain specific, fine grained annotation is generally difficult to acquire. It is our belief that, a neural graph retrieval system should have the capacity to be trained using relevance judgements, similar to classic Information Retrieval systems. Therefore, we do not consider such high-supervision approaches among our baselines.

A.2 Combinatorial approaches in graph and subgraph matching

Combinatorial optimization approaches for graph and subgraph matching are widely studied in the literature. The complexity of graph isomorphism is an open problem, with a well-known quasipolynomial time algorithm (Babai 2016). On the other hand, the subgraph isomorphism problem has been long known to be NP-complete (Cook 1971). The graph matching problem can also be formulated as a quadratic assignment problem (QAP) which is well known to be NP-complete (Garey and Johnson 1979). Classical combinatorial approaches for exact matching are based on backtracking, such as Ullman's Algorithm (Ullmann 1976) and VF2 (Cordella et al. 2004). These suffer from exponential time and memory requirements. Additionally, such methods are unable to learn from any given data distribution. Rather than test for (sub)graph isomorphism in a given pair of graphs, our focus is on *ranking* a number of corpus graphs in terms of *how close* they are to having subgraphs isomorphic to the query graph.

A.3 Graph representation learning

In recent years, there have been a series of formulations (Gilmer et al. 2017; Hamilton, Ying, and Leskovec 2017; Kipf and Welling 2016; Veličković et al. 2017; Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016; Zhang and Chen 2018) which aim to distill sparse, high dimensional neighborhood information into denser, low dimensional compact representation vectors. They are often used to compute similarity between graphs in vector spaces (Li et al. 2019; Bai et al. 2019). Apart from graph matching, they are also used for a wide variety of downstream applications, *e.g.*, link prediction (Zhang and Chen 2018), node classification (Grover and Leskovec 2016), etc.

B Connection between the optimization (6) and the linear assignment problem (9)

The optimization $\min_{\boldsymbol{P}} \sum_{i,j} \left[\left(\boldsymbol{R}_q - \boldsymbol{P}\boldsymbol{R}_c \right)_+ \right]_{i,j}$ is equivalent to the following optimization problem.

$$\min_{\boldsymbol{P},\boldsymbol{\zeta}\geq 0} \sum_{i,j} \boldsymbol{\zeta}_{i,j} \qquad \text{subject to,} \quad \boldsymbol{\zeta}_{i,j} \geq (\boldsymbol{R}_q - \boldsymbol{P}\boldsymbol{R}_c)_{i,j} \tag{22}$$

Given the Lagrangian multiplier matrices $\boldsymbol{C} \geq 0$ and $\boldsymbol{D} \geq 0$, the dual of the above optimization problem (22) becomes:

$$\max_{\boldsymbol{C}_{i,j}\geq 0, \boldsymbol{D}\geq 0} \; \min_{\boldsymbol{P},\boldsymbol{\zeta}\geq 0} \; \sum_{i,j} \boldsymbol{\zeta}_{i,j} + \boldsymbol{C}_{i,j}(\boldsymbol{R}_q - \boldsymbol{P}\boldsymbol{R}_c - \zeta)_{i,j} - \boldsymbol{D}_{i,j}\boldsymbol{\zeta}_{i,j} \tag{23}$$

Differentiating w.r.t $\boldsymbol{\zeta}_{i,j}$, we have

$$1 - \boldsymbol{C}_{i,j} - \boldsymbol{D}_{i,j} = 0 \Rightarrow \boldsymbol{C}_{i,j} + \boldsymbol{D}_{i,j} = 1 \Rightarrow \boldsymbol{C}_{i,j} \in [0,1] \text{ since } \boldsymbol{D}_{i,j} \geq 0 \tag{24}$$

Thus we have,

$$\max_{\boldsymbol{C}\in[0,1]} \min_{\boldsymbol{P}} \sum_{i,j} \boldsymbol{C}_{i,j}(\boldsymbol{R}_q - \boldsymbol{P}\boldsymbol{R}_c)_{i,j} = \max_{\boldsymbol{C}\in[0,1]} \min_{\boldsymbol{P}} \text{Trace}\left[\boldsymbol{C}^\top \left(\boldsymbol{R}_q - \boldsymbol{P}\boldsymbol{R}_c\right)\right] \tag{25}$$

which is a linear assignment problem.

C Dataset preparation

We gather six datasets from the repository of graphs maintained by TUDatasets (Morris et al. 2020). Their characteristics are summarized in Table 7.

For our subgraph matching task, we sample the query and corpus graphs from each of the TUDataset graphs using a breadth first search (BFS) based technique as proposed in Lou et al. (2020). To sample any graph G, we first sample a node u and then perform a random BFS traversal centered around that u till $|V| \in [7, 15]$ nodes have been selected. We extract the subgraph induced by V and set it as G. Having independently sampled all query and corpus graphs using this procedure, we obtain the ground truth subgraph isomorphism relevance labels, by using the Networkx implementation of the VF2 algorithm (Hagberg,

Swart, and S Chult 2008; Lou et al. 2020). Table 8 summarizes the statistics of these sampled datasets, which are used for our experiments. Each of these datasets in Table 8 contains 100 query graphs and 800 corpus graphs. Out of the 80,000 query corpus pairs thus generated, we find that the pos-to-neg relevance ratio takes values between 0.21:1 to 0.26:1 across the different datasets. For the experiments reported in main, we have split the 100 query graphs into 60% training, 15% validation and 25% test folds.

Dataset	No. of graphs	Avg. no of nodes	Avg no. of edges
PTC-FM	349	14.11	14.48
PTC-FR	351	14.56	15.00
PTC-MM	336	13.97	14.32
PTC-MR	344	14.29	14.69
AIDS	2000	15.69	16.20
MUTAG	188	17.93	19.79

Table 7: Statistics of the raw datasets collected from TUDatasets (Morris et al. 2020).

Dataset	Avg. $\|V_q\|$	Avg. $\|E_q\|$	Avg. $\|V_c\|$	Avg. $\|E_c\|$	$\|\{y(G_q,G_c)=1\}\|$	$\|\{y(G_q,G_c)=-1\}\|$	$\frac{\|\{y(G_q,G_c)=1\}\|}{\|\{y(G_q,G_c)=-1\}\|}$
PTC-FM	8.96	8.45	13.24	13.20	14790	65210	0.23
PTC-FR	9.16	8.68	13.32	13.37	13975	66025	0.21
PTC-MM	9.10	8.55	13.29	13.22	15939	64061	0.25
PTC-MR	9.04	8.58	13.32	13.30	14231	65769	0.22
MUTAG	8.87	8.97	13.34	13.76	16420	63580	0.26
AIDS	8.40	7.89	13.12	13.04	13901	66099	0.21

Table 8: Statistics of the sampled subgraphs used in our experiments.

D Implementation details for all methods

D.1 Implementation details of ISONET

ISONET has three modules: (1) The neural alignment module F_θ as described in Eq. (10), (2) the asymmetric hinge scoring module for subgraph matching, as described in Eq. (7) and (3) the encoder module parameterized by ϕ, used for obtain edge representation matrices $\boldsymbol{R}$ for all graphs. In the following, we describe the specifications of each of the modules in detail, beginning with the node features $\boldsymbol{x}$.

Specification of $\boldsymbol{x}_\bullet$. We aim to tackle the problem of feature agnostic subgraph matching. Hence we set the initial node features to the same value $\boldsymbol{x}_\bullet = [1]$. This ensures that all the node embeddings start with the same features and structurally isomorphic nodes end up having the similar embeddings.

Specification of neural alignment module F_θ. F_θ as described in Eq. (10) realizes Gumbel-Sinkhorn operator on the product $LRL_\theta(\boldsymbol{R}_q) \cdot LRL_\theta(\boldsymbol{R}_c)$. Here, LRL_θ is a three layer neural network comprising of one linear, one ReLU and and one linear layer, having latent feature dimension 16. In all cases we use 20 Sinkhorn Operator iterations, with a temperature of 0.1. The input of F_θ is a pair of edge embedding matrices: $\boldsymbol{R}_q$ for query graph $G_q = (V_q, E_q)$ and $\boldsymbol{R}_c$ for corpus graph $G_c = (V_c, E_c)$. $\boldsymbol{R}_q$ is padded with $|E_c| - |E_q|$ rows of zeros, indicating dummy edges. The output of F_θ is a doubly stochastic matrix of dimension equal to $|E_c|$.

Specification of scoring module. We implement the distance function of Eq. (7), which computes an asymmetric distance measure over a proposed alignment of $\boldsymbol{R}_q$ and $\boldsymbol{R}_c$. The negative of this distance measure is used as a similarity score for ranking loss.

Specification of graph embedding network. Here, we follow the same architecture proposed in the graph embedding model of GMN-embed. More in details, INIT_ϕ consists of a single linear layer generating 10 dimensional node features; MSG_ϕ consists of a single linear layer which takes as input a pair of node embedding vectors and outputs a message vector of dimension 20; AGGR_ϕ aggregates the message vectors incoming to any node using a simple sum, thus generating the aggregate message $\overline{\boldsymbol{r}}_\bullet$ for the node; and, COMB_ϕ is a gated recurrent unit (GRU), with the aggregated message $\overline{\boldsymbol{r}}_\bullet$ treated as the input to the GRU. The current node embedding is treated as the hidden state of the GRU, which gets updated conditioned on the input $\overline{\boldsymbol{r}}_\bullet$.

Here, a single propagation step comprises of executing the MSG_ϕ, AGGR_ϕ and COMB_ϕ modules. In AGGR, we use $K = 5$ propagation steps, which encapsulates structural information from the $K-$hop local neighborhood around a node into its embedding.

D.2 Implementation details of baselines

Here, we specify the implementation details for each of the baselines.

RWKernel and SPKernel. We use the Grakel implementation.(Siglidis et al. 2020)[4]

GMN-embed and GMN-match. We use the official Pytorch implementation (Li et al. 2019).[5]

NeuroMatch. We use the official implementation by the authors. For a fair comparison with all other algorithms, we do not provide anchor node annotations. The input graphs were the same as ISONET, and the output similarity scores were given to the ranking module for training.[6]

SimGNN. We use the official Pytorch implementation (Bai et al. 2019).[7]

GOTSim. We use the official Pytorch implementation (Doan et al. 2021). However, the original implementation operates on one graph pair at a time. We found that to be prohibitively slow for processing our larger training dataset. So we have implemented a faster version which operates on batched inputs, while ensuring consistency of outputs with respect to original implementation. Still, we were only able to achieve moderate speedup ($\approx 3\times$), since GOTSim uses a black box combinatorial solver, which requires input to given one graph pair at a time.[8]

GraphSim. We leverage the implementation provided by the GOTSim authors. However, we had to implement a batched version for speedup. This time we were able to achieve a more significant speedup ($\approx 8\times$).[9]

Moreover, we ensure a fair comparison of our method against the baselines as follows: (1) Input node features provided to all baselines are exactly the same as to ISONET. No other (node or edge) features or labels are used at any point. (2) All baselines are trained using the exact same ranking loss as described in Eq. (19). More details regarding hyperparameter settings are provided in Appendix D.3. (3) All baselines are trained with an early stopping setup similar to ISONET, as the stopping criteria of the training process. More details regarding hyperparameters and choice of optimizer are provided in Appendix D.3 (4) The node and graph level embedding dimensions, for the baselines, are the same as in case of ISONET (namely, 10). (5) To the extent possible, we ensure that, for all baselines, the the number of trainable parameters are close to that in ISONET. We present parameter counts for all baselines in Table 9.

	Parameter Count
GraphSim	2067
GOTSim	304
SimGNN	1671
GMN-embed	1750
GMN-match	2050
NeuroMatch	3463
ISONET	2028

Table 9: Number of parameters in each trainable model.

D.3 Hyperparameter details

In all experiments, we use an early stopping criteria based on validation MAP. Here, we set the "patience" parameter as 50 epochs, *i.e.*, if the validation MAP does not improve continuously for 50 epochs, we stop training and take the best model obtained so far. We train all models using the ranking loss defined in Eq. (19) with Adam optimizer, learning rate 10^{-3} and weight decay 5×10^{-4}. We checked the performance of ISONET and baselines for margins ranging over $[1, 0.001]$ and generally found the performance to peak at 0.5. Only for GOTsim, peak performance was observed at margin 0.1. For all models and datasets, we report the best results observed across the margins.

D.4 Evaluation Metrics

Following standard practice in Information Retrieval systems, we evaluate the performance of a graph retrieval system in terms of Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR). Given a query graph G_q, the retrieval system is expected to return a ranked list of corpus graphs. The average precision AP_q is computed against the ground truth relevance labels. Also, rank_q is the rank of the topmost relevant corpus graph, for the given query graph. Subsequently, we compute MAP and MRR scores as follows:

$$\text{MAP} = \frac{1}{|Q|}\sum_{q \in Q} \text{AP}_q, \quad \text{MRR} = \frac{1}{|Q|}\sum_{q \in Q} \frac{1}{\text{rank}_q} \tag{26}$$

[4]https://github.com/ysig/GraKeL

[5]https://github.com/Lin-Yijie/Graph-Matching-Networks

[6]https://github.com/snap-stanford/neural-subgraph-learning-GNN

[7]https://github.com/benedekrozemberczki/SimGNN

[8]https://github.com/khoadoan/GraphOTSim

[9]https://github.com/khoadoan/GraphOTSim

D.5 Infrastructure details

We implement ISONET using Python 3.9.6 and PyTorch 1.9.0. The experiments were run on servers equipped with Xeon E5-2620 2.10GHz CPUs, Nvidia Quadro RTX 8000-48 GB GPUs, and Nvidia Titan Xp-12 GB GPUs.

E Additional experiments

E.1 MRR: ISONET vs node alignment

	PTC-FR	PTC-FM	PTC-MM	PTC-MR	MUTAG	AIDS
Node-align (Node loss)	0.940	0.980	0.973	**1.000**	**0.980**	0.964
Node-align (Edge loss)	0.922	0.940	0.933	0.861	0.924	0.926
ISONET	**0.960**	**1.000**	**0.980**	0.970	0.940	**1.000**

Table 10: Performance comparison, in terms of MRR, of ISONET against its two alternatives: **Node-align (Node loss)** and **Node-align (Edge loss)**, that are obtained by replacing its edge alignment network with two node alignment networks, across all datasets. In the first alternative, we compute the score $d_{\theta,\phi}(G_c \,|\, G_q) = \sum_{i,j}[(\boldsymbol{H}_q - \boldsymbol{S}\boldsymbol{H}_c)_+]_{i,j}$. In the second alternative, we compute $d_{\theta,\phi}(G_c \,|\, G_q) = \sum_e[(\boldsymbol{L}_q - \boldsymbol{S}\boldsymbol{L}_c\boldsymbol{S}^\top)_+]_e$, where $\boldsymbol{L}(e) = \boldsymbol{A}(e) \cdot [\text{FF}(\boldsymbol{r}_e)]$. Numbers in **bold** indicate the best performer. The results are similar to the observations of Table 3, *i.e.*, edge alignment is more effective than node alignment.

E.2 MRR: Symmetric vs Asymmetric scores

	ISONET		GMN-embed		GMN-match	
	Symm	Asym	Symm	Asym	Symm	Asym
PTC-FR	0.940	**0.960**	**0.960**	0.908	0.953	0.940
PTC-FM	0.932	**1.000**	0.841	0.968	0.920	0.890
PTC-MM	0.946	**0.980**	0.901	0.907	0.946	0.973
PTC-MR	**1.000**	0.970	0.933	0.935	0.960	0.914
MUTAG	0.907	0.940	**0.947**	0.884	0.933	0.916
AIDS	0.980	**1.000**	**1.000**	0.973	0.953	0.944

Table 11: Performance comparison in terms of mean reciprocal rank (MRR) between asymmetric (7) and symmetric scoring module, measured using MRR - for ISONET, GMN-match and GMN-embed, across all datasets. Numbers in **bold** indicate the best performer. The results are similar to the observations of Table 4, *i.e.*, asymmetric scoring is more effective than symmetric scoring.

E.3 Visualization of isomorphic subgraph pairs

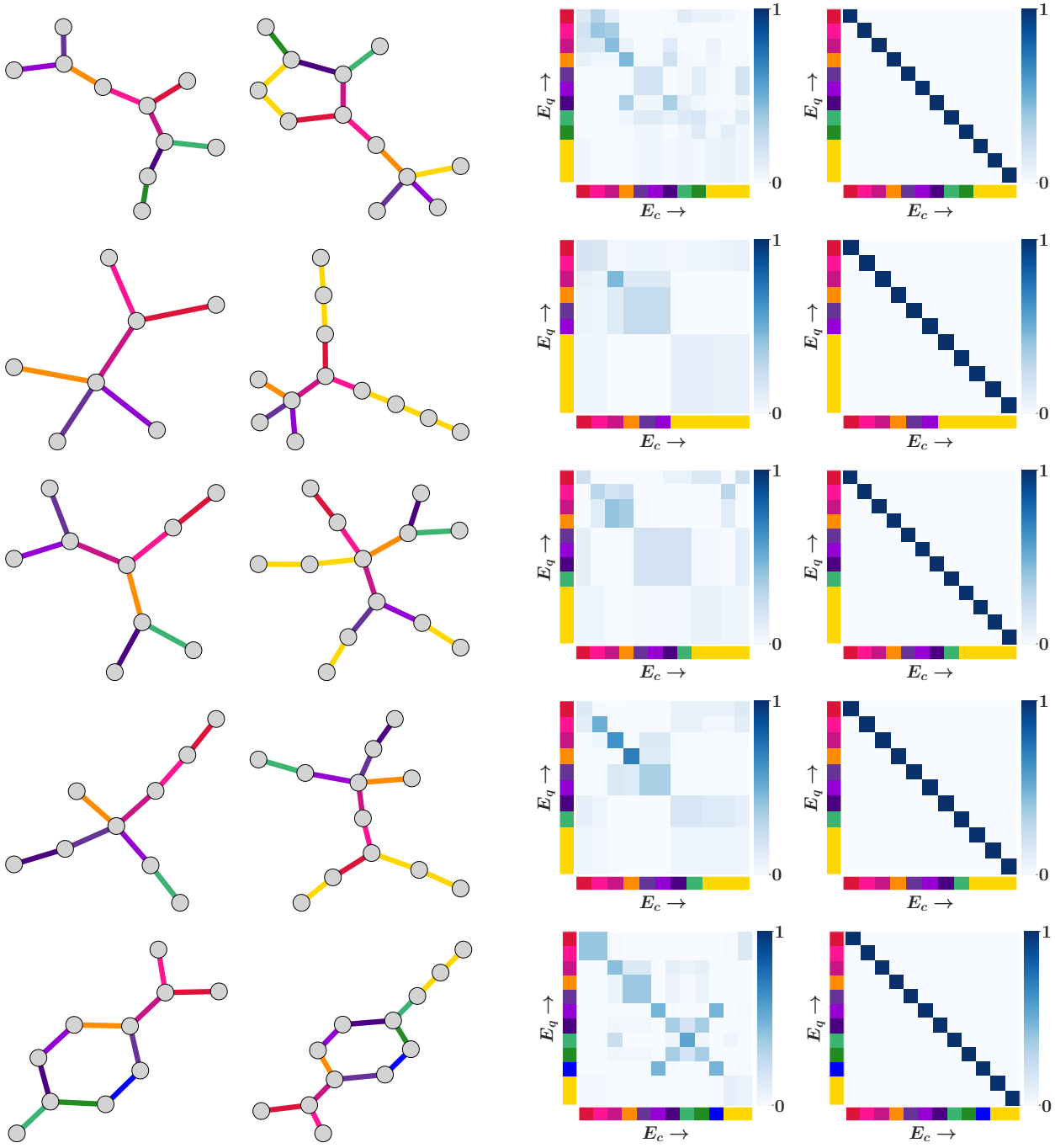
In Figure 12, we show example graph pairs with seeded isomorphisms, where ISONET is able to propose perfect edge alignments. Each edge is color-coded; the same colors are used in the edge-to-edge correspondence matrix, where the heatmap shows the extent of match found by ISONET. First we show the two graphs G_q and G_c. Then we show the soft correspondence matrix. Finally, we apply a Hungarian assignment algorithm to show a 'hard' correspondence matrix in the rightmost column. Edges are ordered such that the planted isomorphism shows up as a diagonal matrix, if ISONET is successful. We make the following observations:

(1) Padded (dummy) edges of the query graph are correctly mapped to the corpus edges which are not part of the underlying subgraph isomorphic to the query graph. This is a significant accomplishment, given that such a subset selection task is combinatorially expensive, with $\binom{N}{k}$ possible outcomes.

(2) Having identified this underlying relevant corpus subgraph, ISONET is also able to find a perfect edge alignment between the query graph and this relevant subgraph of corpus graph. Specifically, it satisfies the node consistency criteria that if any two edges of the query graph share a node, then their matched edges on the corpus side should also have a node in common. This can be thought of as a logical counterpart to the usual requirement of node alignments honoring edge correspondence.

(3) The off-diagonal entries in the soft correspondence matrix are significant — they correspond to edge pairs with locally similar neighborhoods.

(4) Sometimes, in case of perfect local symmetries (consider axially flipping the "benzene ring" in the last example), the Hungarian assignment will achieve the same objective for all members of the symmetric assignment group. Running it with different initializations will sometimes flip the edge assignment accordingly.



(a) G_q (b) G_c (c) $\boldsymbol{P}_{\text{soft}}$ (d) $\boldsymbol{P}_{\text{hard}}$

Figure 12: Example graph pairs such that the query graph G_q (Panel (a)) is an isomorphic subgraph of the corpus graph (Panel (b)). The matrix $\boldsymbol{P}_{\text{soft}}$ (Panel (c)) denotes the soft edge alignments predicted by IsoNet. The right-most permutation matrix $\boldsymbol{P}_{\text{hard}}$ (Panel (c)) is the hard alignment obtained by using a Hungarian solver on top of the soft assignment matrix.

E.4 Additional experiments on large datasets

Dataset preparation. We also evaluate ISONET's performance against the baselines, on a larger query set of 300 graphs. The statistics for this larger dataset are summarized in Table 13. Note that, we trained our retrieval systems on the original smaller dataset. Therefore, the current evaluation will provide a more aggressive test of the underlying graph retrieval model.

Dataset	Avg. $\|V_q\|$	Avg. $\|E_q\|$	Avg. $\|V_c\|$	Avg. $\|E_c\|$	$\|\{y(G_q, G_c) = 1\}\|$	$\|\{y(G_q, G_c) = -1\}\|$	$\frac{\|\{y(G_q,G_c)=1\}\|}{\|\{y(G_q,G_c)=-1\}\|}$
PTC-FM	8.78	8.30	13.24	13.20	48910	191090	0.26
PTC-FR	8.93	8.46	13.32	13.37	46805	193195	0.24
PTC-MM	8.85	8.40	13.29	13.22	48475	191525	0.25
PTC-MR	8.71	8.20	13.32	13.30	47252	192748	0.25
MUTAG	9.08	9.11	13.34	13.76	51265	188735	0.27
AIDS	8.57	8.15	13.12	13.04	46109	193891	0.24

Table 13: Statistics of the sampled subgraphs for the larger dataset with 300 query graphs.

MAP and MRR analysis. Next, we compare our method against all baselines for this larger dataset. Table 14 summarizes the results, which reveal similar observations as in Table 2 in the main paper. Moreover, in all datasets other than MUTAG, the MAP gain achieved by ISONET against the second best baseline is statistically significant (Welch's t-test, $p < 0.001$). In terms of MRR, ISONET shows statistically significant performance gain for all datasets except PTC-MR and MUTAG (Welch's t-test, $p < 0.001$).

	Mean Average Precision (MAP)					
	PTC-FR	PTC-FM	PTC-MM	PTC-MR	MUTAG	AIDS
SPKernel	0.35 ± 0.01	0.38 ± 0.01	0.38 ± 0.01	0.37 ± 0.01	0.42 ± 0.01	0.36 ± 0.01
RWKernel	0.20 ± 0.00	0.21 ± 0.00	0.21 ± 0.00	0.21 ± 0.01	0.21 ± 0.01	0.19 ± 0.00
GraphSim	0.39 ± 0.01	0.39 ± 0.01	0.41 ± 0.01	0.45 ± 0.01	0.38 ± 0.01	0.37 ± 0.01
GOTSim	0.49 ± 0.01	0.60 ± 0.01	0.47 ± 0.01	0.57 ± 0.01	0.59 ± 0.01	0.55 ± 0.01
SimGNN	0.40 ± 0.01	0.58 ± 0.01	0.40 ± 0.01	0.45 ± 0.01	0.50 ± 0.01	0.37 ± 0.01
GMN-embed	0.82 ± 0.01	0.81 ± 0.01	0.79 ± 0.01	0.83 ± 0.01	0.87 ± 0.01	0.79 ± 0.01
GMN-match	0.84 ± 0.01	0.83 ± 0.01	0.86 ± 0.01	0.86 ± 0.01	0.91 ± 0.01	0.82 ± 0.01
NeuroMatch	0.74 ± 0.01	0.74 ± 0.01	0.78 ± 0.01	0.67 ± 0.01	0.81 ± 0.01	0.76 ± 0.01
ISONET	$\mathbf{0.91 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$	$\mathbf{0.93 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$
	Mean Reciprocal Rank (MRR)					
	PTC-FR	PTC-FM	PTC-MM	PTC-MR	MUTAG	AIDS
SPKernel	0.66 ± 0.02	0.62 ± 0.02	0.69 ± 0.02	0.64 ± 0.02	0.74 ± 0.02	0.60 ± 0.02
RWKernel	0.50 ± 0.02	0.46 ± 0.02	0.50 ± 0.02	0.52 ± 0.02	0.20 ± 0.02	0.41 ± 0.02
GraphSim	0.89 ± 0.01	0.61 ± 0.02	0.84 ± 0.02	0.88 ± 0.02	0.70 ± 0.02	0.57 ± 0.02
GOTSim	0.81 ± 0.02	0.87 ± 0.01	0.71 ± 0.02	0.89 ± 0.01	0.80 ± 0.02	0.83 ± 0.02
SimGNN	0.78 ± 0.02	0.83 ± 0.02	0.79 ± 0.02	0.82 ± 0.02	0.66 ± 0.02	0.81 ± 0.02
GMN-embed	0.98 ± 0.01	0.90 ± 0.01	0.92 ± 0.01	0.90 ± 0.01	0.96 ± 0.01	0.93 ± 0.01
GMN-match	0.96 ± 0.01	0.94 ± 0.01	0.98 ± 0.01	$\mathbf{0.98 \pm 0.00}$	$\mathbf{0.99 \pm 0.01}$	0.95 ± 0.01
NeuroMatch	0.96 ± 0.01	0.96 ± 0.01	0.98 ± 0.01	0.90 ± 0.01	0.90 ± 0.02	0.98 ± 0.01
ISONET	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.98 \pm 0.01}$	$\mathbf{0.99 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$

Table 14: Graph retrieval performance measured in terms of mean average precision (MAP) (top half) and mean reciprocal rank (MRR) (bottom half) for the larger datasets. Here, we also report the standard error across all the queries along with the mean. We consider all methods, *i.e.*, ISONET and all the state-of-the-art baselines, *viz.*, Shortest path kernel (SPKernel), Random Walk kernel (RWKernel) (Vishwanathan et al. 2010), GraphSim (Bai et al. 2020), GOTSim (Doan et al. 2021), SimGNN (Bai et al. 2019), GMN-embed, GMN-match (Li et al. 2019) and Neuromatch (Lou et al. 2020). The experimental setup is same as in Table 2. Here, we observe that superior performance of ISONET also holds true for this larger dataset.

Drill down analysis. We evaluate the performance of ISONET against baseline retrieval models, by comparing per query graph AP scores for the larger query set of 300 graphs. In particular, for each query graph, we measure ISONET's gain or loss with respect to the three top performing baselines GMN-embed, GMN-match and NeuroMatch.

PTC-FR: ISONET outperforms GMN-embed for 92.3%, GMN-match for 84.0% and Neuromatch for 86.0% of the query graphs.

PTC-FM: ISONET outperforms GMN-embed for 96.7%, GMN-match for 94.7% and Neuromatch for 93.7% of the query graphs.

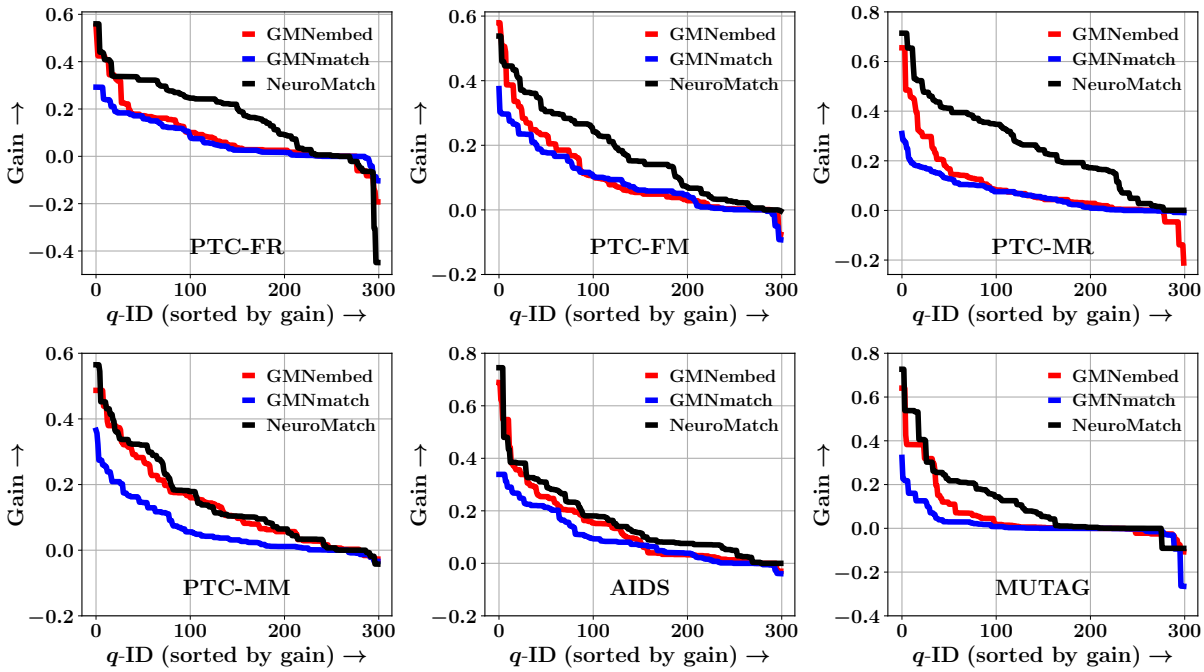


Figure 15: Evaluation of per-query performance of IsoNet against top performing baselines. We plot AP(IsoNet)-AP(Baseline), which measures the gain (above x axis) or loss (below x-axis), in decreasing order of magnitude. The plot for GMN-match lies closest to the x axis, thus indicating it to be the second best performer across all datasets.

PTC-MR: IsoNet outperforms GMN-embed for 92.3%, GMN-match for 81.0% and Neuromatch for 100% of the query graphs.
PTC-MM: IsoNet outperforms GMN-embed for 93.3%, GMN-match for 83.0% and Neuromatch for 93.7% of the query graphs.
AIDS: IsoNet outperforms GMN-embed for 98.3%, GMN-match for 84.3% and Neuromatch for 94.3% of the query graphs.
MUTAG: IsoNet outperforms GMN-embed for 67.3%, GMN-match for 51.0% and Neuromatch for 86.0% of the query graphs.

References

Babai, L. 2016. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, 684–697.

Bai, Y.; Ding, H.; Bian, S.; Chen, T.; Sun, Y.; and Wang, W. 2019. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 384–392.

Bai, Y.; Ding, H.; Gu, K.; Sun, Y.; and Wang, W. 2020. Learning-based efficient graph similarity computation via multi-scale convolutional set matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3219–3226.

Bernard, F.; Theobalt, C.; and Moeller, M. 2018. Ds*: Tighter lifting-free convex relaxations for quadratic matching problems. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4310–4319.

Cereto-Massagué, A.; Ojeda, M. J.; Valls, C.; Mulero, M.; Garcia-Vallvé, S.; and Pujadas, G. 2015. Molecular fingerprint similarity search in virtual screening. *Methods*, 71: 58–63.

Cho, M.; Lee, J.; and Lee, K. M. 2010. Reweighted random walks for graph matching. In *European conference on Computer vision*, 492–505. Springer.

Cook, S. A. 1971. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, 151–158.

Cordella, L. P.; Foggia, P.; Sansone, C.; and Vento, M. 2004. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence*, 26(10): 1367–1372.

Cuturi, M. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26: 2292–2300.

Doan, K. D.; Manchanda, S.; Mahapatra, S.; and Reddy, C. K. 2021. Interpretable Graph Similarity Computation via Differentiable Optimal Alignment of Node Embeddings.

Fey, M.; Lenssen, J. E.; Morris, C.; Masci, J.; and Kriege, N. M. 2020. Deep graph matching consensus. *arXiv preprint arXiv:2001.09621*.

Garey, M. R.; and Johnson, D. S. 1979. *Computers and intractability*, volume 174. freeman San Francisco.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, 1263–1272. PMLR.

Grohe, M.; Rattan, G.; and Woeginger, G. J. 2018. Graph similarity and approximate isomorphism. *43rd International Symposium on Mathematical Foundations of Computer Science*.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*.

Hagberg, A.; Swart, P.; and S Chult, D. 2008. Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1025–1035.

Joachims, T. 2002. Optimizing Search Engines Using Clickthrough Data. In *SIGKDD Conference*, 133–142. ACM.

Johnson, J.; Krishna, R.; Stark, M.; Li, L.-J.; Shamma, D.; Bernstein, M.; and Fei-Fei, L. 2015. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3668–3678.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Lai, A.; and Hockenmaier, J. 2017. Learning to predict denotational probabilities for modeling entailment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, 721–730.

Leordeanu, M.; and Hebert, M. 2005. A spectral technique for correspondence problems using pairwise constraints.

Li, Y.; Gu, C.; Dullien, T.; Vinyals, O.; and Kohli, P. 2019. Graph matching networks for learning the similarity of graph structured objects. In *International conference on machine learning*, 3835–3845. PMLR.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Ling, X.; Wu, L.; Wang, S.; Pan, G.; Ma, T.; Xu, F.; Liu, A. X.; Wu, C.; and Ji, S. 2021. Deep Graph Matching and Searching for Semantic Code Retrieval. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(5): 1–21.

Liu, L.; Hughes, M. C.; Hassoun, S.; and Liu, L.-P. 2021. Stochastic Iterative Graph Matching. *arXiv preprint arXiv:2106.02206*.

Liu, T.-Y. 2009. Learning to Rank for Information Retrieval. In *Foundations and Trends in Information Retrieval*, volume 3, 225–331. Now Publishers.

Lou, Z.; You, J.; Wen, C.; Canedo, A.; Leskovec, J.; et al. 2020. Neural Subgraph Matching. *arXiv preprint arXiv:2007.03092*.

Mena, G.; Belanger, D.; Linderman, S.; and Snoek, J. 2018. Learning latent permutations with gumbel-sinkhorn networks. *arXiv preprint arXiv:1802.08665*.

Morris, C.; Kriege, N. M.; Bause, F.; Kersting, K.; Mutzel, P.; and Neumann, M. 2020. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*.

Nowak, A.; Villar, S.; Bandeira, A. S.; and Bruna, J. 2018. Revised note on learning quadratic assignment with graph neural networks. In *2018 IEEE Data Science Workshop (DSW)*, 1–5. IEEE.

Ohlrich, M.; Ebeling, C.; Ginting, E.; and Sather, L. 1993. Subgemini: Identifying subcircuits using a fast subgraph isomorphism algorithm. In *Proceedings of the 30th International Design Automation Conference*, 31–37.

Peng, Y.; Choi, B.; and Xu, J. 2021. Graph Edit Distance Learning via Modeling Optimum Matchings with Constraints.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *KDD*, 701–710.

Qu, J.; Ling, H.; Zhang, C.; Lyu, X.; and Tang, Z. 2021. Adaptive Edge Attention for Graph Matching with Outliers.

Schellewald, C.; and Schnörr, C. 2005. Probabilistic subgraph matching based on convex relaxation. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 171–186. Springer.

Siglidis, G.; Nikolentzos, G.; Limnios, S.; Giatsidis, C.; Skianis, K.; and Vazirgiannis, M. 2020. GraKeL: A Graph Kernel Library in Python. *J. Mach. Learn. Res.*, 21: 54–1.

Tan, H.-R.; Wang, C.; Wu, S.-T.; Wang, T.-Q.; Zhang, X.-Y.; and Liu, C.-L. 2021. Proxy Graph Matching with Proximal Matching Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 9808–9815.

Ullmann, J. R. 1976. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, 23(1): 31–42.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Vendrov, I.; Kiros, R.; Fidler, S.; and Urtasun, R. 2015. Order-embeddings of images and language. *ICLR 2016*.

Vishwanathan, S. V. N.; Schraudolph, N. N.; Kondor, R.; and Borgwardt, K. M. 2010. Graph kernels. *Journal of Machine Learning Research*, 11: 1201–1242.

Wang, R.; Yan, J.; and Yang, X. 2019. Learning combinatorial embedding networks for deep graph matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3056–3065.

Wong, E. K. 1992. Model matching in robot vision by subgraph isomorphism. *Pattern Recognition*, 25(3): 287–303.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Yan, X.; Yu, P. S.; and Han, J. 2004. Graph indexing: A frequent structure-based approach. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 335–346.

Yu, T.; Wang, R.; Yan, J.; and Li, B. 2019. Learning deep graph matching with channel-independent embedding and hungarian attention. In *International conference on learning representations*.

Yu, T.; Wang, R.; Yan, J.; and Li, B. 2021. Deep Latent Graph Matching. In *International Conference on Machine Learning*, 12187–12197. PMLR.

Yu, T.; Yan, J.; Wang, Y.; Liu, W.; and Li, B. 2018. Generalizing graph matching beyond quadratic assignment model. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 861–871.

Zanfir, A.; and Sminchisescu, C. 2018. Deep learning of graph matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2684–2693.

Zhang, M.; and Chen, Y. 2018. Link prediction based on graph neural networks. In *NeurIPS*.

Zhao, K.; Tu, S.; and Xu, L. 2021. IA-GM: A Deep Bidirectional Learning Method for Graph Matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 3474–3482.