A. PROOFS OF SUBGRAPH PROPERTIES

The paper introduces the following observations that justify the use of order embeddings in subgraph matching.

Transitivity. Suppose that G_1 is a subgraph of G_2 with bijection f mapping all nodes from G_1 to a subset of nodes in G_2, and G_2 is a subgraph of G_3 with bijection g. Let v_1, v_2, v_3 be anchor nodes of G_1, G_2, G_3 respectively. By definition of anchored subgraph, $f(v_1) = v2$ and $g(v_2) = v_3$. Then the composition $g \circ f$ is a bijection. Moreover, $g \circ f(v_1) = g(v_2) = v_3$, where Therefore G_1 is a subgraph of G_3, and thus the transitivity property.

This corresponds to the transitivity of order embedding.

Anti-symmetry. Suppose that G_1 is a subgraph of G_2 with bijection f, and G_2 is a subgraph of G_1 with bijection g. Let $|V_1|$ and $|V_2|$ be the number of nodes in G_1 and G_2 respectively. By definition of subgraph isomorphism, G_1 is a subgraph of G_2 implies that $|V_1| \leq |V_2|$. Similarly, G_2 is a subgraph of G_1 implies $|V_2| \leq |V_1|$. Hence $|V_1| = |V_2|$. The mapping between all nodes in G_1 and G_2 is bijective. By definition of isomorphism, G_1 and G_2 are graph-isomorphic.

This corresponds to the anti-symmetry of order embedding.

Intersection. By definition, if G_3 is a common subgraph of G_1, G_2, the G_3 is a subgraph of both G_1 and G_2. Since a trivial node is a subgraph of any graph, there is always a non-empty intersection set between two graphs.

Correspondingly, if $z_3 \preceq z_1$ and $z_3 \preceq z_2$, then $z_3 \preceq \min\{z_1, z_2\}$. Here $\min$ denotes the element-wise minimum of two embeddings. Note that the order embedding z_1 and z_2 are positive, and therefore $\min\{z_1, z_2\}$ is another valid order embedding, corresponding to the non-empty intersection set between two graphs.

Note that this paper assumes the frequent motifs are connected graphs. And thus it also assumes that all neighborhoods in a given datasets are connected and contain at least 2 nodes (an edge). This is a reasonable assumption since we can remove isolated nodes from the datasets, as connected motifs of size k ($k > 1$) can never contain isolated nodes. In this case, the trivial intersection corresponds to a graph of 2 nodes and 1 edge.

For all datasets, we randomly sample connected subgraph queries as test sets, with diameter less than 8, a mild assumption since most of the graph datasets have diameter less than 8.

B. ORDER EMBEDDING COMPOSITION

We can show that the order constraints in Equation 1 hold under the composition of multiple message passing layers of the GNN, assuming simple GNN models such as in paper "Simplifying Graph COnvolutional Networks" and "Scalable Inception Graph Networks".

Suppose that we use a k-layer GNN to encode nodes u and v in the search and query graphs respectively. If the k-hop neighborhood of u is a subgraph of the k-hop neighborhood of v, then $\forall s \in \mathcal{N}_v$, $\exists t \in \mathcal{N}_u$ such that the $(k-1)$-hop neighborhood of s must be a subgraph of the $(k-1)$-hop neighborhood of t. Neighborhoods of u's neighbors are subgraphs of a subset of the $(k-1)$-hop neighborhoods of v's neighbors.

Consequently, we can guarantee the following observation with order embeddings:

Observation 2. *Suppose that all GNN embeddings at layer $k-1$ satisfy order constraints after transformation. Then when using sum-based neighborhood aggregation, the GNN embeddings at layer k also satisfy the order constraints.*

After applying linear transformations and non-linearities in the GNN at layer $k-1$, if the order embedding of all neighbors of node v are no greater than that of the corresponding matched nodes in the target graph (*i.e.* satisfy the order constraint), then when summing the order embeddings of neighbors to compute embedding of v at layer k, it is guaranteed that node v also satisfies the order constraint at layer k. This corresponds to the property of composition of subgraphs into larger subgraphs.

In other GNN architectures, such properties do not necessarily hold, due to the presence of transformation and non-linearity at each convolution layer. However, this provides another alignment between

Model	Accuracy
SAGE (2-LAYER, 32-DIM, DROPOUT=0.2)	77.5
SAGE (6-LAYER, 32-DIM, DROPOUT=0.2)	85.3
SAGE (8-LAYER, 64-DIM, DROPOUT=0.2)	86.3
GCN (6-LAYER, 64-DIM, DROPOUT=0.2)	69.9
GCN (9-LAYER, 128-DIM, DROPOUT=0.2)	82.3
GIN (4-LAYER, 32-DIM, DROPOUT=0.2)	81.0
GIN (4-LAYER, 64-DIM, DROPOUT=0)	87.0
GIN (8-LAYER, 64-DIM, DROPOUT=0)	88.4
SAGE (4-LAYER, 64-DIM, DROPOUT=0)	87.6
SAGE (8-LAYER, 64-DIM, DROPOUT=0)	89.4
SAGE (12-LAYER, 64-DIM, DROPOUT=0)	90.5
SAGE (8-LAYER, 64-DIM, DROPOUT=0, SKIP-LAYER)	91.5

Table 4: The accuracy (unit: 0.01) for matching on the ENZYMES dataset for different model configurations.

the order embedding objective and the subgraph matching task in terms of growing neighborhoods, and motivates the use of curriculum learning for this task.

C. VOTING PROCEDURE

The voting procedure is used to improve certainty of matched pairs by considering presence of nearby matched pairs in neighborhoods of the matched pairs. The method is motivated by the following observation.

Observation 3. *Let $\mathcal{N}^{(l)}$ denotes the l-hop neighborhood. Then, if $q \in G_Q$ and node $u \in G_T$ match, then for all nodes $i \in N^{(k)}(q)$, $\exists$ node $j \in \mathcal{N}^{(l)}(u), l \leq k$ such that node i and node j match.*

Algorithm 2: NeuroMatch Voting Algorithm

Input: Query node $q \in G_Q$, target node $u \in G_T$. Threshold t for violation below which we predict positive subgraph relation between the neighborhoods of q and u.
Output: Whether the node pair matches.
Compute embeddings for neighbors of q, u within K hops
for hop $k \leq K$ **do**
  for node $i \in N^{(k)}(q)$ **do**
    $m = \min\{E(z_i, z_j) | \forall j \in N^{(k)}(u)\}$
    If $m > t$, **return** False
return True

Since the query graph G_Q is a subgraph of target graph G_T, all paths in G_Q have corresponding paths in G_T. Hence the shortest distance of a node $i \in N^{(k)}(q)$ to q is at most the shortest distance of node $j \in \mathcal{N}^{(l)}(u)$ in G_T, where j is the corresponding node in G_T defined by the subgraph isomorphism mapping. However, the shortest paths are not necessarily of equal lengths, since in G_T there might be additional short-cuts from j to u that do not exist in G_Q.

D. TRAINING DETAILS AND HYPERPARAMETERS

All models are trained on a single GeForce RTX 2080 GPU, and both the heuristics and neural models use a Intel Xeon E7-8890 v3 CPU.

Curriculum training. In each epoch, we iterate over all target graphs in the curriculum and randomly sample one query per target graph. We lower bound the number of iterations per epoch to 64 for datasets that are too small. For the E-R dataset, where we generate neighborhoods at random, and the WN dataset which consists of only a single graph, we use a fixed 64 iterations per epoch. On all datasets except for the E-R dataset, we used 256 target graphs where possible. At training time, we enforce a 3:1 negative to positive ratio in the training examples, which is necessary since in reality there is a heavy skew in the dataset towards negative examples. 10% of the negative examples are hard negatives; among the remaining 90%, half are negative examples drawn from the same target graph as the query, and half are negative examples drawn from different target graphs.

The model is trained with a learning rate of 1×10^{-3} using the Adam optimizer. The learning rate is annealed with a cosine annealer with restarts every 100 epochs. The curriculum starts with 1 target graph with a radius of 1; it is updated every time there are 20 consecutive epochs without an

improvement of more than 0.1. The curriculum update increases the radius of the target graphs by 1 up to a maximum of 4, after which it doubles the number of target graphs for every update up to a maximum of 256. The dataset is regenerated every 50 epochs.

	Predicted +	Predicted −
Positive	68.2	8.3
Negative	70.5	1030.9

Table 5: Average confusion Matrix for matching small queries (size ≤ 7) to all node neighborhoods in the DD dataset.

Hyperparameters. We performed a comprehensive sweep over hyperparameters used in the model. Table 4 shows the effect of hyperparameters and GNN models on the performance, using the NeuroMatch framework. We list the design choices we made that are observed to perform well in both synthetic and real-world datasets:

- Sum aggregation usually works the best, confirming previous theoretical studies (Xu et al., 2018). Both the GraphSAGE and GIN architecture we implemented uses the sum neighborhood aggregation.
- We observe slight improvement in performance when using LeakyReLU instead of ReLU for non-linearity.
- Dropout does not have a significant impact on performance.
- Adding structural features, such as node degree, clustering coefficient, and average path length improves the convergence speed.

Matching query to target graph by aggregating scores. In Problem 1 (Table 2), all methods must make a binary prediction of whether the query is a subgraph of the target graph based on the alignment matrix $\mathcal{A}$ of scores $f(z_q, z_u)$ between all pairs of query and target neighborhoods. In order to aggregate the scores contained in the alignment matrix, we adopt the simple strategy of taking the mean of all entries in the matrix, which we found to outperform the commonly-used Hungarian algorithm on our binary decision task. The exception is FASTPFP, which provides a discrete assignment matrix matching each query node to a target node; for this method, we adopt the following prediction score:

$$\frac{\|A_{\text{pred}} - A_{\text{query}}\|_1}{|V_{\text{query}}|^2} + \frac{X_{\text{pred}} X_{\text{query}}^T}{|V_{\text{query}}|}$$

where A_{pred} is the adjacency matrix of the predicted matched graph, A_{query} is the adjacency matrix of the query graph, $|V_{\text{query}}|$ is the number of nodes in the query and X_{pred} and X_{query} are the feature matrices of nodes in the predicted and query graph, respectively. This score measures the degree to which the predicted and query graph match in terms of topology and node labels, and is based on the loss function used in the paper, but is adapted to compare matchings across varying query and target sizes. In general, we found these aggregation strategies to be effective in our setting containing diverse query and target sizes, but our method is agnostic to such downstream processing of the alignment matrix. In particular, the Hungarian algorithm or other alignment resolution algorithms can still be used with the alignment matrix generated by NeuroMatch, especially when an explicit matching (rather than a binary subgraph prediction) is desired.

For baseline hyperparameters: for ISORANKN, we set $K = 10$, threshold to 1e-4, alpha to 0.9 and the maximum vector length to 1000000. For FASTPFP, we set lambda to 1, alpha to 0.5, and both thresholds to 1e-4.

E. SUBGRAPH MATCHING DATA STATISTICS

E.1. DATASETS

Biology and chemistry datasets. COX2 contains 467 graphs of chemical molecules with an average of 41 nodes and 44 edges each. DD contains 1178 graphs with an average of 284 nodes and 716 edges.

Dataset	ENZYMES	COX2	AIDS	PPI	IMDB-BINARY
IN-DOMAIN	92.9	97.2	94.3	89.9	81.8
TRANSFER	78.9	93.9	92.2	81.0	74.2

Table 6: The AUROC (unit: 0.01) for matching on real datasets, where we either train on the synthetic dataset and test generalization to the real dataset (TRANSFER), or train directly on the dataset that we test on (IN-DOMAIN).

Dataset	COX2	DD	MSRC_21	FIRSTMMDB	ENZYMES	SYNTHETIC
Target size (nodes)	41.6	30.0	79.6	30.0	35.7	30.2
Target size (edges)	43.8	61.2	204.0	49.9	67.2	119.1
Query size (nodes)	22.4	17.8	22.6	17.8	17.4	17.5
Query size (edges)	23.0	34.4	46.3	27.9	29.4	53.4
Query:target size ratio (nodes)	53.8	59.3	28.4	59.3	48.7	57.9

Table 7: Statistics of target and query graphs used in evaluation of Problem 1 (Table 2).

It describes protein structure graphs where nodes are amino acids and edges represent positional proximity. We use node labels for both of the datasets. PPI dataset contains the protein-protein interaction graphs for human tissues. It has 24 graphs corresponding to different PPI networks of different human tissues. In total, there are 56944 nodes and 818716 edges. We do not include node features for PPI networks since the goal is to match various protein interaction patterns without considering the identity of proteins.

MSRC_21 is a semantic image processing dataset introduced in (Winn et al., 2005), containing 563 graph each representing the graphical model of an image. It has an average of 78 nodes and 199 edges.

FIRSTMMDB is a point cloud dataset containing 3d point clouds for various household objects. It tocontains 41 graphs with an average of 1377 nodes and 3074 edges each.

Label imbalance. We performed additional experiments to investigate the confusion matrix for the DD dataset averaged across test queries. Table 5 shows extreme imbalance (subgraphs are rare).

Matching query to target graph. Table 7 shows the statistics of target and query graphs used to evaluate performance on Problem 1 (Table 2).

F. GENERALIZATION AND RUNTIME

F.1. PRETRAINING ON SYNTHETIC DATASET

To demonstrate the use and generalizability of the synthetic dataset, we conducted the experiment where the subgraph matching model is trained only on the synthetic dataset, and is then tested on real-world datasets. Table 6 shows the generalization performance. The first row corresponds to the model performance when trained and tested on the same dataset. The second row corresponds to the model performance when trained on the synthetic dataset, and tested on queries sampled from real-world datasets (listed in each column), Although there is a drop in performance when the model only sees the synthetic dataset, the model is able generalize to a diverse setting of subgraph matching scenarios, in biology, chemistry and social network domains, even out-performing some baseline methods that are specifically trained on the real-world datsets.

However, a shortcoming is that since the synthetic dataset does not contain node features, and real datasets have varying node feature dimensions, the model is only able to consider subgraph matching task that does not take feature into account. Incorporation of feature in transfer learning of subgraph matching remains to be an open problem.

G. COMPARISON TO EXACT AND APPROXIMATE HEURISTICS

G.1. EXACT HEURISTICS METHODS

Exact heuristics such as VF2 and RI algorithms guarantees to make the correct prediction of whether query is a subgraph of the target. However, even for relatively small queries (of size 20), matching is

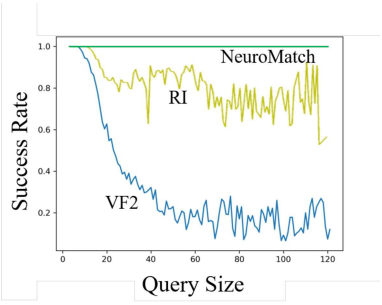

Figure 4: Runtime analysis. Success rate of baseline heuristic matching algorithms (VF2 and RI) for matching in under 20 seconds. NeuroMatch achieves 100% success rate.

costly and can sometimes take unexpectedly long time in the order of hours. As such, these algorithms are not suitable in online or high-throughput scenarios where efficiency is priority.

To demonstrate the runtime efficiency, we show in Figure 4 the success rate of the exact methods, which drop below 60% when the query size is increased to more than 30. In comparison, NeuroMatch always finishes under 0.1 second.

Table 8 shows the runtime comparison between NeuroMatch and the exact baselines considered (VF2 and RI). NeuroMatch achieves 100 times speedup compared to these exact methods.

Moreover, since in practice, it is feasible to pre-train the NeuroMatch model on synthetic datasets, and optionally finetune few epochs on real-world datasets, the training time for model when given a new dataset is also negligible. However, such approach has the limitation that the model cannot account for node categorical features when performing subgraph matching, since the synthetic dataset does not contain any node feature.

Datasets	E-R	MSRC_21	DD
VF2	25.9	19.7	22.8
RI	12.8	7.5	11.0
NEUROMATCH-MLP	0.49	0.48	0.44
NEUROMATCH-ORDER	0.04	0.03	0.03

Table 8: Average runtime (in seconds) comparison between heuristic methods and our method with query size up to 50. NeuroMatch is about 100x faster than alternatives.

G.2. APPROXIMATE HEURISTICS METHODS

Additionally, there have been many works focusing on heuristic methods for motif/subgraph counting (Ribeiro et al., 2019), notable methods include Rand-ESU, MFinder, Motivo, ORCA. However, these works primarily focus on fast enumeration of small motifs typically of size less than 6. In our cases, the size of target and query is much larger (up to hundreds in size), and we do not focus on enumeration of motifs of certain size.

A related line of work is graph matching, or finding an explicit (sub)graph isomorphism mapping between query and target nodes. Methods include convex relaxations (FastPFP, PATH) and spectral approaches (IsoRankN). Such approaches are inherently heuristic-based due to the hardness of approximation of the subgraph matching problem.

H. GNN EXPRESSIVE POWER

Previous works (Xu et al., 2018; Morris et al., 2019) have identified limitations of a class of GNNs. More specifically, GNNs face difficulties when asked to distinguish regular graphs. In this work, we circumvent the problem by distinguishing the anchor node and other nodes in the neighborhood via one-hot encoding (See Section 3.2). The idea is explored in a concurrent work "Identity-aware Graph Neural Networks" (ID-GNNs). It uses Figure 5 to demonstrate the expressive power of ID-GNN, which distinguishes anchor node from other nodes. For example, while d-regular graphs such as 3-cycle and 4-cycle graphs have the same GNN computational graphs, their ID-GNN computational graphs are different, due to identification of anchor nodes via node features. Such modification enables better expressive power than message-passing GNNs such as GIN.

A future direction is to investigate the performance of recently proposed more expressive GNNs (Chen et al., 2019) in the context of subgraph mining. The NeuroMatch framework is general and any GNN can be used in its decoder component, and could benefit from more expressive GNNs.

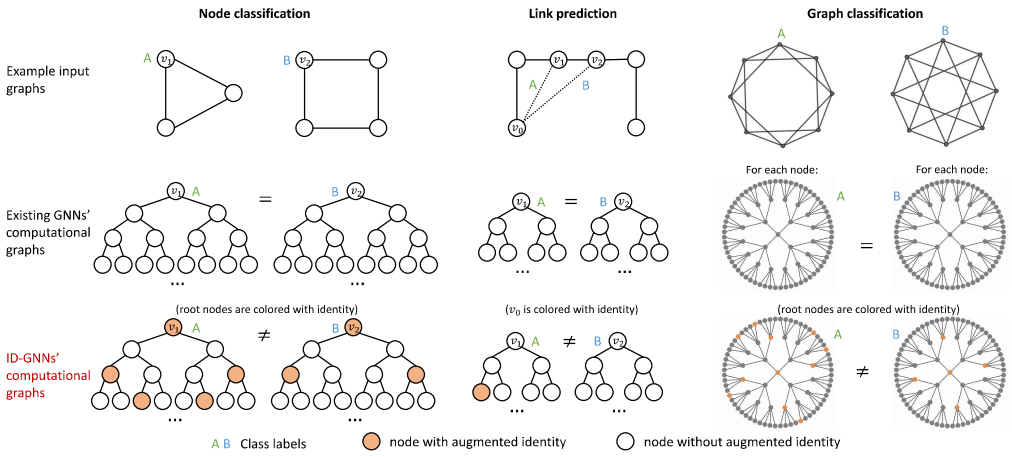

Figure 5: An overview of the proposed ID-GNN model. We consider node, edge and graph level tasks, and assume nodes do not have additional features. Across all examples, the task requires an embedding that allows for the differentiation of the label A vs. B nodes in their respective graphs. However, across all tasks, existing GNNs, regardless of depth, will *always* assign the same embedding to both classes of nodes, because for all tasks the computational graphs are identical. In contrast, the colored computation graphs provided by ID-GNNs allows for clear differentiation between the nodes of class A and class B, as the colored computation graph are no longer identical across all tasks.