Subgraphormer: Unifying Subgraph GNNs and Graph Transformers via Graph Products

Guy Bar-Shalom[1], Beatrice Bevilacqua[2], Haggai Maron[1,3]

Technion[1], Purdue University[2], Nvidia[3]

Introduction

We introduce a novel approach that combines the theoretical foundations of Subgraph-GNNs with the empirical strengths of Graph-Transformers. By establishing a connection between Subgraph-GNNs and the Graph Cartesian Product, we enable the efficient integration of these methodologies.

Goals

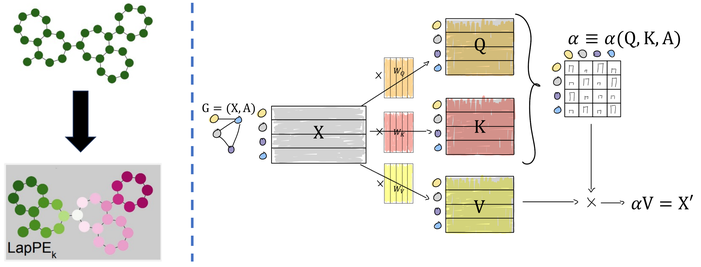
1. Efficiently integrate positional encodings tailored for Subgraph-GNNs.
2. Incorporate attention mechanisms into the structure of Subgraph-GNNs.

Contributions

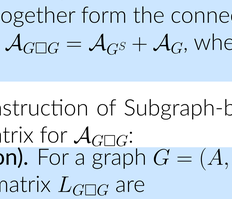
1. Subgraphormer, an architecture that combines the strengths of both transformer-based and Subgraph-based architectures.
2. An observation connecting Subgraph-GNNs to product graphs.
3. A positional encoding scheme tailored to subgraph methods.
4. An empirical study demonstrating significant improvements of Subgraphormer compared to existing baselines.

Preliminaries

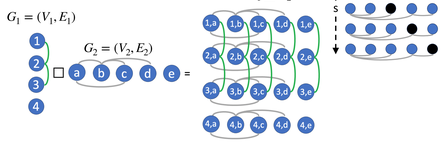
Graph Transformers. Graph Transformers typically employ Graph Positional Encodings, such as Laplacian eigenvectors, alongside the attention mechanism applied to the new node feature matrix.



Subgraph GNNs. Subgraph GNNs represent a graph, G as a multiset or a bag of subgraphs. The bag of subgraphs is generated through the application of a predefined selection policy to the original graph. This bag is processed via an equivariant architecture.



Graph Cartesian Product. The following is a example of the graph Cartesian product between two graphs, G_1, G_2.

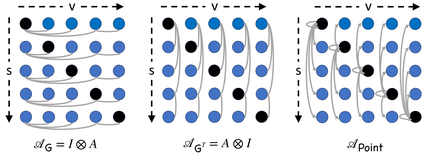


Subgraphormer

The objectives of Subgraphormer can be summarized as follows,

- Integrate attention-based aggregations to Subgraph GNNs.
- Integrate Subgraph-based Positional Encodings.

We build upon the updates used in the maximally expressive Subgraph GNN [1]:



Attention mechanisms can be integrated using RGCN [2], modified to employ attention-based updates.

Subgraph-based PE. The challenges in constructing Subgraph-based PE are,

1. **Adjacency.** What adjacency should be used?
2. **Efficiency.** The adjacency matrices for Subgraph GNNs are in $\mathbb{R}^{n^2 \times n^2}$, making the eigendecomposition of their Laplacian matrix computationally expensive.

A natural solution to (1) is to use the adjacencies $\mathcal{A}_G$ and $\mathcal{A}_{G^T}$, excluding $\mathcal{A}_{\text{Point}}$ since it is asymmetrical and its connectivity is unrelated to the original graph. This brings us to challenge (2).

We present a proposition drawing an analogy between the primary connectivity of the maximally expressive Subgraph GNN and the Cartesian product of graphs.

Proposition (Subgraph GNNs and Cartesian Products). The internal and external subgraph connectivities, $\mathcal{A}_G$ and $\mathcal{A}_{G^S}$, together form the connectivity of the Cartesian product graph $\mathcal{A}_{G \square G}$. Specifically, $\mathcal{A}_{G \square G} = \mathcal{A}_{G^S} + \mathcal{A}_G$, where $\mathcal{A}_{G^S} = A \otimes I$ and $\mathcal{A}_G = I \otimes A$.

The following proposition enables efficient construction of Subgraph-based PE via the eigendecomposition of the Laplacian matrix for $\mathcal{A}_{G \square G}$:

Proposition (Product Graph Eigendecomposition). For a graph $G = (A, X)$, the eigenvectors and eigenvalues of the Laplacian matrix $L_{G \square G}$ are $\{(v_i \otimes v_j, \lambda_i + \lambda_j)\}_{i,j=1}^{n^2}$, where $\{(v_i, \lambda_i)\}_{i=1}^{n}$ are the eigenvectors and eigenvalues of the Laplacian matrix of G.

Subgraphormer handles n^2-node graphs, but its PE only requires the eigendecomposition of the original n-node graph, maintaining a time complexity of $O(k \cdot n^2)$, similar to the original graph's PE calculation.

Experiments

The top three results are reported as First, Second, and Third.

Model ↓ / Dataset →	Param.	ZINC-12k (MAE ↓)	ZINC-Full (MAE ↓)
GIN	500k	0.163 ±0.004	-
GPS	424k	**0.070** ±0.004	-
Graphormer-GD	503k	0.081 ±0.009	0.025 ±0.004
DSS-GNN [5]	100k	0.102 ±0.003	0.029 ±0.003
GNN-SSWL+ [1]	387k	**0.070** ±0.005	**0.022** ±0.001
`Subgraphormer`	293k	**0.067** ±0.007	**0.020** ±0.002
`Subgraphormer + PE`	293k	**0.063** ±0.001	**0.023** ±0.001

Model ↓ / Dataset →		Peptides-func (AP ↑)	Peptides-struct (MAE ↓)
GCN		0.5930 ±0.0023	0.3496 ±0.0013
GatedGCN+RWSE		0.6069 ±0.0035	0.3357 ±0.0006
Transf.+LapPE		0.6326 ±0.0126	0.2529 ±0.0016
SAN+RWSE		**0.6439** ±0.0075	0.2545 ±0.0012
GPS		**0.6535** ±0.0041	**0.2500** ±0.0005
GNN-SSWL+	30%	0.5847 ±0.0050	0.2570 ±0.0006
`Subgraphormer`	30%	**0.6415** ±0.0052	**0.2494** ±0.0020
`Subgraphormer + PE`	30%	0.6373 ±0.0110	**0.2475** ±0.0007

ZINC. On ZINC-12k and ZINC-Full (left table), `Subgraphormer` outperforms both Subgraph GNNs and Graph Transformers with fewer parameters. The attention mechanism and product graph PE boost its performance, making it the top method on ZINC-12k.

Peptides. On Peptides-func and Peptides-struct (right table), `Subgraphormer` shows excellent scalability and captures long-range dependencies. With a 30% sampling ratio, it achieves top performance on Peptides-struct and comparable results on Peptides-func.

References

[1] Bohang Zhang, Guhao Feng, Yiheng Du, Di He, Liwei Wang.
A Complete Expressiveness Hierarchy for Subgraph GNNs via Subgraph Weisfeiler-Lehman Tests.
ICML 2023.

[2] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, Max Welling.
Modeling Relational Data with Graph Convolutional Networks.
ESWC 2018.