Graph Inductive Bias in Transformers without Message Passing

Jiaqing (Steven) Xie
Supervisor: Florian Grötschla
05/03/2024
Individual Modelling

Training LSTM on texts

Training CNN on images

Training **GNN** on graphs
Message Passing Neural Networks

Q: Want to update node 1 in one-hop

1. Compute messages
   \[ M_{12} = \text{Message}(X_1, X_2, e_{12}) \]
   \[ M_{13} = \text{Message}(X_1, X_3, e_{13}) \]
   \[ M_{15} = \text{Message}(X_1, X_5, e_{15}) \]

   Example: \[ M_{12} = W * X_1 + X_2 * \text{MLP}(e_{12}) + b \]

2. Aggregate messages
   Mean aggregation:
   \[ M_{1\text{new}} = \frac{1}{3} \times (M_{12} + M_{13} + M_{15}) \]

3. Update node feature (for node 1)
   Update by MLP / GRU:
   \[ X_{1\text{new}} = W * X_1 + U * M_{1\text{new}} + b, \text{ where } W, U \text{ and } b \text{ are learnable parameters} \]
Unified Encoding Scheme: Transformer

Language Transformer Model (Vaswani 2017 et al.)

Vision Transformer Model (Dosovitskiy 2021 et al.)
Special Graphs

Path Graph

Grid (Lattice) Graph

Example:
I am an ETH student

Example:
split image into 9 patches
Positional Encoding for Special Graphs

Path Graph

Grid (Lattice) Graph

1-dim coordinate system:

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

2-dim coordinate system:

Absolute PE: Sinusoidal

Relative PE:

$$PE_{ij} = PE_i - PE_j$$
Positional Encoding for General Graphs

It is hard to directly observe positional encodings for general graphs!

- There’s no natural Coordinate system for graphs
- Canonical Ordering is limited to planar graphs
- Some solutions: DFS / BFS / Random Walk
Graph Transformers

Family of Graph Transformers (Luis et al. 2024)
Positional Encoding (PE)

- Shortest Path Distance (SPD)
  - Example: for node 1, SPD PE is [0, 0.1, 0.2, 0.5, 0.3, 0.3]

- Eigenvalue Decomposition on Graph Laplacians

- Random Walk:
  - Example: Walk length = 4, starting from node 1: \((1, 3, 4, 2, 6)\) => generate a RW corpus
  - Remember Word2Vec

- Node2Vec: Biased Random Walk
  - Explore more: \((1, 3, 4, 2, 6)\)
  - Return more: \((1, 3, 1, 2, 1)\)
MPNN vs. Global Attention

Propagation in MPNN:
- Complexity: $O(|V|)$ for sparse graphs where $|V| >> |E|$
- Capture neighborhood nodes

Propagation in Global Attention:
- Complexity: $O(|V|^2)$
- Capture all nodes in graph

Additional propagation in graph transformer:
- Complexity: $O(|V|^2)$
- Capture all nodes in graph
Inductive Bias

Informal definition: model’s capability of capturing prior information of data

Model Sets

- **Inductive Bias**
  - **Time Dependence**
    - RNN
  - **Neighbor Aggregation**
    - GNN
  - **Local Connectivity**
    - CNN
Inductive Bias (CNN vs. Transformer)

CNNs serves locality while self-attention layers are global.

Neighbor information is aggregated by the kernel in CNN.

Positions of pixels are unknown for self-attention blocks.

Locality: ✔️

Locality: ❌
Graph Inductive Bias (MPNN vs. Graph Transformer)

MPNNs serves locality while self-attention layers in GT are global.

Neighbor nodes’ information is aggregated by MPNN.

Potential unlinked nodes are supposed to be linked under the settings of graph transformer.

 Locality: ✔

 Locality: ✗
Pros and cons of MPNN and GT

Pros
• MPNN focus on local dependencies. It’s more effective where local graph topology takes matter.
• GT focus on global dependencies. It works well on graph level tasks.

Cons
• MPNN suffers from over-smoothing where all node representations are the same.
• Graph transformers suffer from the missing of graph inductive bias (local topology).
Combine MPNN and GT: GraphGPS

**Good** ☑: Insert MPNN to GT will bring graph inductive bias in GT.

**Bad** ☒: New model inherits oversmoothing from MPNN

Without MPNN in GT? Yes, if PE + GT is as good as MPNN.

GraphGPS Layer (Rampášek et al. 2022)
Overview: GRIT

Overall architecture of GRIT
Transformer with Positional Encoding

$G = (V, E)$

RRWP (Pre-compute) → GRIT Transformer Block $\times N$ → Graph Prediction Head → MSA → DegScaler → BN → FFN → BN → $x_{i}^{\text{out}}$ → $e_{i,j}^{\text{out}}$
Relative Random Walk Probabilities (RRWP)

- RRWP is a positional encoding method for graph
- Define $A$: Adjacency Matrix
- Define $D$: Diagonal Degree Matrix
- Define $M = D^{-1}A$: prob of node i to node j in one RW

Example:

\[
A = \begin{bmatrix}
0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

\[
D = \begin{bmatrix}
3 & 0 & 0 & 0 & 0 & 0 \\
0 & 3 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 \\
\end{bmatrix}
\]

\[
D^{-1} = \begin{bmatrix}
1/3 & 0 & 0 & 0 & 0 & 0 \\
0 & 1/3 & 0 & 0 & 0 & 0 \\
0 & 0 & 1/2 & 0 & 0 & 0 \\
0 & 0 & 0 & 1/2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1/2 \\
0 & 0 & 0 & 0 & 0 & 1/2 \\
\end{bmatrix}
\]

\[
M = \begin{bmatrix}
0 & 0.333 & 0.333 & 0 & 0.333 & 0 \\
0.333 & 0 & 0 & 0.333 & 0 & 0.333 \\
0.5 & 0 & 0 & 0.5 & 0 & 0 \\
0 & 0.5 & 0.5 & 0 & 0 & 0 \\
0.5 & 0 & 0 & 0 & 0 & 0.5 \\
0 & 0.5 & 0 & 0 & 0.5 & 0 \\
\end{bmatrix}
\]
Relative Random Walk Probabilities (RRWP)

- Define RRWP: $P_{ij} = [I_{ij}, M_{ij}, M_{ij}^2, ..., M_{ij}^K]$
- Definition: $M_{ij}^K$: prob of node $i$ to node $j$ by $K$ hops

Example:

Diagram:

$I = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$

$M^3 = \begin{bmatrix}
0.00 & 0.43 & 0.29 & 0.00 & 0.29 & 0.00 \\
0.43 & 0.00 & 0.00 & 0.29 & 0.00 & 0.29 \\
0.43 & 0.00 & 0.00 & 0.35 & 0.00 & 0.22 \\
0.00 & 0.43 & 0.35 & 0.00 & 0.22 & 0.00 \\
0.43 & 0.00 & 0.00 & 0.22 & 0.00 & 0.35 \\
0.00 & 0.43 & 0.22 & 0.00 & 0.35 & 0.00
\end{bmatrix}$

$M = \begin{bmatrix}
0 & 0.333 & 0.333 & 0 & 0.333 & 0 \\
0.333 & 0 & 0 & 0.333 & 0 & 0.333 \\
0.5 & 0 & 0 & 0.5 & 0 & 0 \\
0 & 0.5 & 0.5 & 0 & 0 & 0 \\
0.5 & 0 & 0 & 0 & 0 & 0.5 \\
0 & 0.5 & 0 & 0 & 0.5 & 0
\end{bmatrix}$

$M^2 = \begin{bmatrix}
0.444 & 0 & 0 & 0.278 & 0 & 0.278 \\
0 & 0.444 & 0.278 & 0 & 0.278 & 0 \\
0 & 0.417 & 0.417 & 0 & 0.167 & 0 \\
0.417 & 0 & 0 & 0.417 & 0 & 0.167 \\
0 & 0.417 & 0.167 & 0 & 0.417 & 0 \\
0.417 & 0 & 0 & 0.167 & 0 & 0.417
\end{bmatrix}$
Relative Random Walk Probabilities (RRWP)

- Define RRWP: $P_{ij} = [I_{ij}, M_{ij}, M_{ij}^2, \ldots, M_{ij}^K]$
- Definition: $M_{ij}^K$: prob of $i$ to $j$ by $K$ hops

Example:

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M^3 = \begin{bmatrix} 0.00 & 0.43 & 0.29 & 0.00 & 0.29 & 0.00 \\ 0.43 & 0.00 & 0.00 & 0.29 & 0.00 & 0.29 \\ 0.43 & 0.00 & 0.00 & 0.35 & 0.00 & 0.22 \\ 0.00 & 0.43 & 0.35 & 0.00 & 0.22 & 0.00 \\ 0.43 & 0.00 & 0.00 & 0.22 & 0.00 & 0.35 \\ 0.00 & 0.43 & 0.22 & 0.00 & 0.35 & 0.00 \end{bmatrix}$$

Q: What is $P_{2,3}$ if $K = 3$?

$$P_{2,3} = [0, 0, 0.278, 0]$$
RRWP + MLP is expressive

Expressive power:
Using RRWP could approximate other PE(s) with MLP

\[(a) \quad \text{MLP}(P)_{ij} \approx \text{SPD}_{K-1}(i, j)\]

\[(b) \quad \text{MLP}(P) \approx \sum_{k=0}^{K-1} \theta_k (D^{-1}A)^k\]

\[(c) \quad \text{MLP}(P) \approx \theta_0 I + \theta_1 A,\]
Test graph isomorphism: WL-test

\[ c^{(k+1)}(v) = \text{HASH}(c^{(k)}(v), \{c^{(k)}(u)\}_{u \in N(v)}) \]

Hash Table
2: \{1, \{1, 1\}\}
3: \{1, \{1, 1, 1\}\}
4: \{2, \{2, 3\}\}
5: \{2, \{3, 3\}\}
6: \{3, \{2, 2, 3\}\}
Test graph isomorphism: WL-test

\[ c^{(k+1)}(v) = \text{HASH}\left( c^{(k)}(v), \{ c^{(k)}(u) \}_{u \in N(v)} \right) \]

**C_2**

Graph 1:
- 6
- 5
- 4

Graph 2:
- 6
- 5
- 4

**C_3**

Graph 1:
- 9
- 8
- 7

Graph 2:
- 9
- 8
- 7

**L_3**

Graph 1:
- 6
- 5
- 4

Graph 2:
- 6
- 5
- 4

---

**Hash Table**

- 2: \{1, \{1, 1\}\}
- 3: \{1, \{1, 1, 1\}\}
- 4: \{2, \{2, 3\}\}
- 5: \{2, \{3, 3\}\}
- 6: \{3, \{2, 2, 3\}\}
- 7: \{4, \{4, 6\}\}
- 8: \{5, \{6, 6\}\}
- 9: \{6, \{4, 5, 6\}\}

---

**Careful!**

The result is only guaranteed to be correct if the test fails. If the coloring matches, we do not have a guarantee for isomorphism.
GD-WL: General WL-test with PE

• Intuition: coloring with SPD

\[ \chi^t_G(v) = \text{hash}(\{(d_G(v, u), \chi^{t-1}_G(u)) : u \in V\}) \]

• SPD fails with GD-WL (edge weight = 1)

• Reason: for each node, k-hop neighbor array is fixed:
  
  \[
  (3, 6, 6, 3, 1)
  \]
  
  \[
  \Rightarrow \{1, 1, 1\}, \{2, 2, 2, 2, 2\}, \{3, 3, 3, 3, 3\}, \{4, 4, 4\}, \{5\}\]

Rethinking the expressive power of gnnns via graph biconnectivity, Zhang et al., ICLR 2023
RRWP is more expressive than SPD

RRWP succeeds with GD-WL
Recall: GRIT

Overall architecture of GRIT
Transformer with Positional Encoding

Internal architecture of GRIT
Flexible Attention

Initialization

\[ x_i = [x'_i \parallel P_{i,i}] \in \mathbb{R}^{d_h+K} \]
\[ e_{i,j} = [e'_{i,j} \parallel P_{i,j}] \in \mathbb{R}^{d_e+K} \]
Flexible Attention

Initialization

\[ \mathbf{x}_i = [\mathbf{x}'_i \parallel \mathbf{P}_{i,i}] \in \mathbb{R}^{d_h + K} \]
\[ \mathbf{e}_{i,j} = [\mathbf{e}'_{i,j} \parallel \mathbf{P}_{i,j}] \in \mathbb{R}^{d_e + K} \]

Attention Computation:

\[ \hat{\mathbf{e}}_{i,j} = \sigma \left( \rho \left( (\mathbf{W}^Q \mathbf{x}_i + \mathbf{W}^K \mathbf{x}_j) \odot \mathbf{W}^E \mathbf{e}_{i,j} + \mathbf{W}^E_b \mathbf{e}_{i,j} \right) \right) \in \mathbb{R}^{d'} \]
\[ \alpha_{ij} = \text{Softmax}_{j \in V} (\mathbf{W}^A \hat{\mathbf{e}}_{i,j}) \in \mathbb{R}, \]
\[ \hat{\mathbf{x}}_i = \sum_{j \in V} \alpha_{ij} \cdot (\mathbf{W}^V \mathbf{x}_j + \mathbf{W}^E \hat{\mathbf{e}}_{i,j}) \in \mathbb{R}^d, \]

Recall MPNN:

\[ \mathbf{x}'_i = \Theta \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j \cdot h_\Theta (\mathbf{e}_{i,j}) \]

Is it a MPNN? No, we need to compute attention for each pair of nodes.
Injecting Degree Information

• Degree information injection:

\[ x_{i}^{\text{out}}' := x_{i}^{\text{out}} \odot \theta_1 + (\log(1 + d_i) \cdot x_{i}^{\text{out}} \odot \theta_2) \in \mathbb{R}^d \]

• Why we need degree scaler?
  ➢ Attention is innately invariant to node degrees (mean-aggr in GNN)
    Therefore, it reduces expressive power
  ➢ Adding degree information will introduce inductive bias
Injecting Degree Information

• Degree information injection:

\[ x_{i}^{\text{out}}' := x_{i}^{\text{out}} \odot \theta_1 + (\log(1 + d_i) \cdot x_{i}^{\text{out}} \odot \theta_2) \in \mathbb{R}^d \]

• Why we need degree scaler?
  ➢ Attention is innately invariant to node degrees (mean-aggr in GNN)
    Therefore, it reduces expressive power
  ➢ Adding degree information will introduce inductive bias

• BatchNorm is favored over LayerNorm
  ➢ LayerNorm would cancel out the effect brought by degree scaler.
Experiment: Baselines

➢ SOTA GT: GraphGPS

➢ Other Graph Transformers:
  □ SAN, Graphormer, K-Subgraph SAT, EGT, Graphormer-URPE, Graphormer-GD

➢ SOTA GNN:
  □ CIN, CRaW1, GIN-AK+

➢ Other GNNs:
  □ GIN, GAT, GatedGCN, GatedGCN-LSPE, PNA, DGN, GSN
Experiment: Overview of Benchmarks

**Task type:**
- PATTERN, CLUSTER: node classification (inductive)
- Others: graph classification / graph regression

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Graphs</th>
<th>Avg. # nodes</th>
<th>Avg. # edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZINC(-full)</td>
<td>12,000 (250,000)</td>
<td>23.2</td>
<td>24.9</td>
</tr>
<tr>
<td>MNIST</td>
<td>70,000</td>
<td>70.6</td>
<td>564.5</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>60,000</td>
<td>117.6</td>
<td>941.1</td>
</tr>
<tr>
<td>PATTERN</td>
<td>14,000</td>
<td>118.9</td>
<td>3,039.3</td>
</tr>
<tr>
<td>CLUSTER</td>
<td>12,000</td>
<td>117.2</td>
<td>2,150.9</td>
</tr>
<tr>
<td>Peptides-func</td>
<td>15,535</td>
<td>150.9</td>
<td>307.3</td>
</tr>
<tr>
<td>Peptides-struct</td>
<td>15,535</td>
<td>150.9</td>
<td>307.3</td>
</tr>
<tr>
<td>PCQM4Mv2</td>
<td>3,746,620</td>
<td>14.1</td>
<td>14.6</td>
</tr>
</tbody>
</table>
Benchmark 1: Common Benchmarks for GT(s)

- ZINC: molecule dataset
- MNIST, CIFAR10: image classification datasets
- PATTERN, CLUSTER: synthetic datasets sampled from Stochastic Block Model

MNIST superpixels dataset from the “Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs” paper, containing 70,000 graphs with 75 nodes each. Every graph is labeled by one of 10 classes.
GRIT has on average better or on par performance when it is compared with GraphGPS.
GRIT has overwhelming advantages when it is compared with GNNs.

<table>
<thead>
<tr>
<th>Model</th>
<th>ZINC</th>
<th>MNIST</th>
<th>CIFAR10</th>
<th>PATTERN</th>
<th>CLUSTER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE↓</td>
<td>Accuracy↑</td>
<td>Accuracy↑</td>
<td>Accuracy↑</td>
<td>Accuracy↑</td>
</tr>
<tr>
<td>GCN</td>
<td>0.367 ± 0.011</td>
<td>90.705 ± 0.218</td>
<td>55.710 ± 0.381</td>
<td>71.892 ± 0.334</td>
<td>68.498 ± 0.976</td>
</tr>
<tr>
<td>GIN</td>
<td>0.526 ± 0.051</td>
<td>96.485 ± 0.252</td>
<td>55.255 ± 1.527</td>
<td>85.387 ± 0.136</td>
<td>64.716 ± 1.553</td>
</tr>
<tr>
<td>GAT</td>
<td>0.384 ± 0.007</td>
<td>95.535 ± 0.205</td>
<td>64.223 ± 0.455</td>
<td>78.271 ± 0.186</td>
<td>70.587 ± 0.447</td>
</tr>
<tr>
<td>GatedGCN</td>
<td>0.282 ± 0.015</td>
<td>97.340 ± 0.143</td>
<td>67.312 ± 0.311</td>
<td>85.568 ± 0.088</td>
<td>73.840 ± 0.326</td>
</tr>
<tr>
<td>GatedGCN-LSPE</td>
<td>0.090 ± 0.001</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>PNA</td>
<td>0.188 ± 0.004</td>
<td>97.94 ± 0.12</td>
<td>70.35 ± 0.63</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>DGN</td>
<td>0.108 ± 0.003</td>
<td>—</td>
<td>72.838 ± 0.417</td>
<td>86.680 ± 0.034</td>
<td>—</td>
</tr>
<tr>
<td>GSN</td>
<td>0.101 ± 0.010</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>CIN</td>
<td>0.079 ± 0.006</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>CRaW1</td>
<td>0.085 ± 0.004</td>
<td>97.944 ± 0.050</td>
<td>69.013 ± 0.259</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>GIN-AK+</td>
<td>0.080 ± 0.001</td>
<td>—</td>
<td>72.19 ± 0.13</td>
<td>86.850 ± 0.057</td>
<td>—</td>
</tr>
<tr>
<td>SAN</td>
<td>0.139 ± 0.006</td>
<td>—</td>
<td>—</td>
<td>86.581 ± 0.037</td>
<td>76.691 ± 0.65</td>
</tr>
<tr>
<td>Graphormer</td>
<td>0.122 ± 0.006</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>K-Subgraph SAT</td>
<td>0.094 ± 0.008</td>
<td>—</td>
<td>—</td>
<td>86.848 ± 0.037</td>
<td>77.856 ± 0.104</td>
</tr>
<tr>
<td>EGT</td>
<td>0.108 ± 0.009</td>
<td>98.173 ± 0.087</td>
<td>68.702 ± 0.409</td>
<td>86.821 ± 0.020</td>
<td>79.232 ± 0.348</td>
</tr>
<tr>
<td>Graphormer-URPE</td>
<td>0.086 ± 0.007</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Graphormer-GD</td>
<td>0.081 ± 0.009</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>GPS</td>
<td>0.070 ± 0.004</td>
<td>98.051 ± 0.126</td>
<td>72.298 ± 0.356</td>
<td>86.685 ± 0.059</td>
<td>78.016 ± 0.180</td>
</tr>
<tr>
<td>GRIT (ours)</td>
<td>0.059 ± 0.002*</td>
<td>98.108 ± 0.111</td>
<td>76.468 ± 0.881*</td>
<td>87.196 ± 0.076*</td>
<td>80.026 ± 0.277*</td>
</tr>
</tbody>
</table>
Benchmark 2: Long Range Graph Benchmark

- Peptides: Amino acid datasets
- Peptides-func: 10-task multi-label classification
- Peptides-struct: 11-task regression
- Long range dataset => **Transformer** captures long range information => **GTs** are better

<table>
<thead>
<tr>
<th>Model</th>
<th>Peptides-func</th>
<th>Peptides-struct</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AP↑</td>
<td>MAE↓</td>
</tr>
<tr>
<td>GCN</td>
<td>0.5930 ± 0.0023</td>
<td>0.3496 ± 0.0013</td>
</tr>
<tr>
<td>GINE</td>
<td>0.5498 ± 0.0079</td>
<td>0.3547 ± 0.0045</td>
</tr>
<tr>
<td>GatedGCN</td>
<td>0.5864 ± 0.0035</td>
<td>0.3420 ± 0.0013</td>
</tr>
<tr>
<td>GatedGCN+RWSE</td>
<td>0.6069 ± 0.0035</td>
<td>0.3357 ± 0.0006</td>
</tr>
<tr>
<td>Transformer+LapPE</td>
<td>0.6326 ± 0.0126</td>
<td>0.2529 ± 0.0016</td>
</tr>
<tr>
<td>SAN+LapPE</td>
<td>0.6384 ± 0.0121</td>
<td>0.2683 ± 0.0043</td>
</tr>
<tr>
<td>SAN+RWSE</td>
<td><strong>0.6439 ± 0.0075</strong></td>
<td>0.2545 ± 0.0012</td>
</tr>
<tr>
<td>GPS</td>
<td><strong>0.6535 ± 0.0041</strong></td>
<td><strong>0.2500 ± 0.0012</strong></td>
</tr>
<tr>
<td>GRIT (ours)</td>
<td><strong>0.6988 ± 0.0082</strong>*</td>
<td><strong>0.2460 ± 0.0012</strong>*</td>
</tr>
</tbody>
</table>
## Benchmark 3: Open Graph Benchmark

- **PCQM4Mv2 (OGB)**
  - Large Scale Graph Datasets (over 3,000,000 graphs)
  - GRIT has on par performance with GraphGPS
  - GRIT has less parameters than GraphGPS

<table>
<thead>
<tr>
<th>Method</th>
<th>Model</th>
<th>Valid. (MAE ↓)</th>
<th># Param</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MPNNs</strong></td>
<td>GCN</td>
<td>0.1379</td>
<td>2.0M</td>
</tr>
<tr>
<td></td>
<td>GCN-virtual</td>
<td>0.1153</td>
<td>4.9M</td>
</tr>
<tr>
<td></td>
<td>GIN</td>
<td>0.1195</td>
<td>3.8M</td>
</tr>
<tr>
<td></td>
<td>GIN-virtual</td>
<td>0.1083</td>
<td>6.7M</td>
</tr>
<tr>
<td><strong>Graph</strong></td>
<td>GRPE</td>
<td>0.0890</td>
<td>46.2M</td>
</tr>
<tr>
<td><strong>Transformers</strong></td>
<td>Graphormer</td>
<td>0.0864</td>
<td>48.3M</td>
</tr>
<tr>
<td></td>
<td>TokenGT (ORF)</td>
<td>0.0962</td>
<td>48.6M</td>
</tr>
<tr>
<td></td>
<td>TokenGT (Lap)</td>
<td>0.0910</td>
<td>48.5M</td>
</tr>
<tr>
<td></td>
<td>GPS-small</td>
<td>0.0938</td>
<td>6.2M</td>
</tr>
<tr>
<td></td>
<td>GPS-medium</td>
<td><strong>0.0858</strong></td>
<td>19.4M</td>
</tr>
<tr>
<td></td>
<td>GRIT (ours)</td>
<td><strong>0.0859</strong></td>
<td>16.6M</td>
</tr>
</tbody>
</table>
## Ablation Study: Architectural Design Choices

<table>
<thead>
<tr>
<th>ZINC</th>
<th>MAE ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRIT (ours)</td>
<td>0.059 ± 0.002</td>
</tr>
<tr>
<td>- Remove degree scaler</td>
<td>0.076 ± 0.002</td>
</tr>
<tr>
<td>- Remove the update of RRWP</td>
<td>0.066 ± 0.005</td>
</tr>
<tr>
<td>- Global-attn. → Sparse-attn.</td>
<td>0.066 ± 0.002</td>
</tr>
<tr>
<td>- Degree scaler → Degree encoding</td>
<td>0.072 ± 0.005</td>
</tr>
<tr>
<td>- GRIT-attn. → Graphormer-attn.</td>
<td>0.117 ± 0.028</td>
</tr>
<tr>
<td>- RRWP → RWSE</td>
<td>0.081 ± 0.010</td>
</tr>
<tr>
<td>- RRWP → SPDPE</td>
<td>0.067 ± 0.002</td>
</tr>
</tbody>
</table>

![Diagram of the model architecture](img)
Opinion: Future works

Advantages:

➢ Fewer params compared to other GTs
➢ Importance of positional encodings (GRIT)

Disadvantages:

➢ Complexity of attention: $O(|V|^2)$
➢ Upper bound on expressive power
Conclusion

➢ Design choices for including graph inductive bias in GT (PE / MPNN)

➢ RRWP encodings are expressive

➢ RRWP initialization is more expressive than SPD under GD-WL tests

➢ GRIT is new SOTA graph transformer which excludes message passing
Any Questions
Others: Expressive power of GNN

GIN is as powerful as WL test

\[ c^{(k+1)}(v) = MLP_{\theta}((1 + \epsilon) \cdot MLP_{\psi}(c^{(k)}(v)) + \sum_{u \in N(v)} MLP_{\psi}(c^{(k)}(u))) \]

\[ c^{(k+1)}(v) = \text{HASH}(c^{(k)}(v), \{c^{(k)}(v)\}_{u \in N(v)}) \]
Others: Graphormer Module

Do Transformers Really Perform Bad for Graph Representation?, Ying et al., NeurIPS 2021
Others: Graph Transformer w/ Both Attention

- Structure-Aware

Structure-Aware Transformer for Graph Representation Learning, Chen et al., ICML 2022
Table 12. Runtime and GPU memory for SAN (Kreuzer et al., 2021), GraphGPS (Rampášek et al., 2022) and GRIT (Ours) on ZINC with batch size 32. The timing is conducted on a single NVIDIA V100 GPU and 20 threads of Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz.

<table>
<thead>
<tr>
<th>ZINC</th>
<th>SAN</th>
<th>GraphGPS</th>
<th>GRIT (Ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAE ↓</td>
<td>0.139 ± 0.006</td>
<td>0.070 ± 0.004</td>
<td>0.059 ± 0.002</td>
</tr>
<tr>
<td>PE Precompute-time</td>
<td>10 sec</td>
<td>11 sec</td>
<td>11 sec</td>
</tr>
<tr>
<td>GPU Memory</td>
<td>2291 MB</td>
<td>1101 MB</td>
<td>1865 MB</td>
</tr>
<tr>
<td>Training time</td>
<td>57.9 sec/epoch</td>
<td>24.3 sec/epoch</td>
<td>29.4 sec/epoch</td>
</tr>
</tbody>
</table>
## Others: Detailed dataset descriptions

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Graphs</th>
<th>Avg. # nodes</th>
<th>Avg. # edges</th>
<th>Directed</th>
<th>Prediction level</th>
<th>Prediction task</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZINC(-full)</td>
<td>12,000 (250,000)</td>
<td>23.2</td>
<td>24.9</td>
<td>No</td>
<td>graph</td>
<td>regression</td>
<td>Mean Abs. Error</td>
</tr>
<tr>
<td>MNIST</td>
<td>70,000</td>
<td>70.6</td>
<td>564.5</td>
<td>Yes</td>
<td>graph</td>
<td>10-class classif.</td>
<td>Accuracy</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>60,000</td>
<td>117.6</td>
<td>941.1</td>
<td>Yes</td>
<td>graph</td>
<td>10-class classif.</td>
<td>Accuracy</td>
</tr>
<tr>
<td>PATTERN</td>
<td>14,000</td>
<td>118.9</td>
<td>3,039.3</td>
<td>No</td>
<td>inductive node</td>
<td>binary classif.</td>
<td>Weighted Accuracy</td>
</tr>
<tr>
<td>CLUSTER</td>
<td>12,000</td>
<td>117.2</td>
<td>2,150.9</td>
<td>No</td>
<td>inductive node</td>
<td>6-class classif.</td>
<td>Accuracy</td>
</tr>
<tr>
<td>Peptides-func</td>
<td>15,535</td>
<td>150.9</td>
<td>307.3</td>
<td>No</td>
<td>graph</td>
<td>10-task classif.</td>
<td>Avg. Precision</td>
</tr>
<tr>
<td>Peptides-struct</td>
<td>15,535</td>
<td>150.9</td>
<td>307.3</td>
<td>No</td>
<td>graph</td>
<td>11-task regression</td>
<td>Mean Abs. Error</td>
</tr>
<tr>
<td>PCQM4Mv2</td>
<td>3,746,620</td>
<td>14.1</td>
<td>14.6</td>
<td>No</td>
<td>graph</td>
<td>regression</td>
<td>Mean Abs. Error</td>
</tr>
</tbody>
</table>
## Others: Hyper-parameter settings

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>ZINC/ZINC-full</th>
<th>MNIST</th>
<th>CIFAR10</th>
<th>PATTERN</th>
<th>CLUSTER</th>
</tr>
</thead>
<tbody>
<tr>
<td># Transformer Layers</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>Hidden dim</td>
<td>64</td>
<td>52</td>
<td>52</td>
<td>64</td>
<td>48</td>
</tr>
<tr>
<td># Heads</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Dropout</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.01</td>
</tr>
<tr>
<td>Attention dropout</td>
<td>0.2</td>
<td>0.5</td>
<td>0.5</td>
<td>0.2</td>
<td>0.5</td>
</tr>
<tr>
<td>Graph pooling</td>
<td>sum</td>
<td>mean</td>
<td>mean</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>PE dim (RW-steps)</td>
<td>21</td>
<td>18</td>
<td>18</td>
<td>21</td>
<td>32</td>
</tr>
<tr>
<td>PE encoder</td>
<td>linear</td>
<td>linear</td>
<td>linear</td>
<td>linear</td>
<td>linear</td>
</tr>
<tr>
<td>Batch size</td>
<td>32/256</td>
<td>16</td>
<td>16</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
<td>0.0005</td>
<td>0.0005</td>
</tr>
<tr>
<td># Epochs</td>
<td>2000</td>
<td>200</td>
<td>200</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td># Warmup epochs</td>
<td>50</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Weight decay</td>
<td>$10^{-5}$</td>
<td>$10^{-5}$</td>
<td>$10^{-5}$</td>
<td>$10^{-5}$</td>
<td>$10^{-5}$</td>
</tr>
<tr>
<td># Parameters</td>
<td>473,473</td>
<td>102,138</td>
<td>99486</td>
<td>477,953</td>
<td>432,206</td>
</tr>
</tbody>
</table>
Benchmark 3: Large Dataset

- ZINC-full Dataset
  - 250,000 molecule graphs
  - Higher order GNNs are included in baselines
  - Positional encoding enhanced GNNs are also included

<table>
<thead>
<tr>
<th>Method</th>
<th>Model</th>
<th>ZINC-full (MAE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPNNs</td>
<td>GIN</td>
<td>0.088 ± 0.002</td>
</tr>
<tr>
<td></td>
<td>GraphSAGE</td>
<td>0.126 ± 0.003</td>
</tr>
<tr>
<td></td>
<td>GAT</td>
<td>0.111 ± 0.002</td>
</tr>
<tr>
<td></td>
<td>GCN</td>
<td>0.113 ± 0.002</td>
</tr>
<tr>
<td></td>
<td>MoNet</td>
<td>0.090 ± 0.002</td>
</tr>
<tr>
<td>Higher-order GNNs</td>
<td>δ-2-GNN</td>
<td>0.042 ± 0.003</td>
</tr>
<tr>
<td></td>
<td>δ-2-LGNN</td>
<td>0.045 ± 0.006</td>
</tr>
<tr>
<td>PE-GNN</td>
<td>SignNet</td>
<td><strong>0.024 ± 0.003</strong></td>
</tr>
<tr>
<td>Graph Transformers</td>
<td>Graphormer</td>
<td>0.052 ± 0.005</td>
</tr>
<tr>
<td></td>
<td>Graphormer-URPE</td>
<td>0.028 ± 0.002</td>
</tr>
<tr>
<td></td>
<td>Graphormer-GD</td>
<td><strong>0.025 ± 0.004</strong></td>
</tr>
<tr>
<td></td>
<td>GRIT (ours)</td>
<td><strong>0.023 ± 0.001</strong></td>
</tr>
</tbody>
</table>
Ablation Study 2: Parameter K of RRWP

![Graph showing Mean Absolute Error (MAE) vs. K]
## Others: Hyper-parameter settings

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Peptides-func</th>
<th>Peptides-struct</th>
<th>PCQM4Mv2</th>
</tr>
</thead>
<tbody>
<tr>
<td># Transformer Layers</td>
<td>4</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>Hidden dim</td>
<td>96</td>
<td>96</td>
<td>256</td>
</tr>
<tr>
<td># Heads</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Dropout</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>Attention dropout</td>
<td>0.5</td>
<td>0.5</td>
<td>0.1</td>
</tr>
<tr>
<td>Graph pooling</td>
<td>mean</td>
<td>mean</td>
<td>mean</td>
</tr>
<tr>
<td>PE dim (walk-step)</td>
<td>17</td>
<td>24</td>
<td>16</td>
</tr>
<tr>
<td>PE encoder</td>
<td>linear</td>
<td>linear</td>
<td>linear</td>
</tr>
<tr>
<td>Batch size</td>
<td>32</td>
<td>32</td>
<td>256</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0002</td>
</tr>
<tr>
<td># Epochs</td>
<td>200</td>
<td>200</td>
<td>150</td>
</tr>
<tr>
<td># Warmup epochs</td>
<td>5</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Weight decay</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td># Parameters</td>
<td>443,338</td>
<td>438,827</td>
<td>15.3M</td>
</tr>
</tbody>
</table>
GRIT attention is successful at matching both the sparsity pattern and attention magnitudes of the target (far left)

Visualization of learned attention scores for the synthetic experiment on learning to attend to (k = 1, 2, 3)-hop neighbors
Others: Synthetic Experiments