

Structure Recognition with Graph Neural Networks

A project for the lab-course
Advanced Projects in Computational Physics 2



Table of contents

1. Theoretical Background

1.1 A Few More Words on Message Passing

1.2 Percolation

2. Goals

3. Results

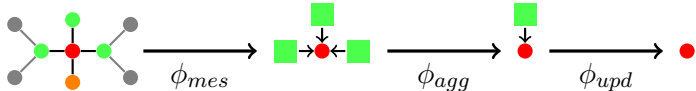
3.1 Classification of Bravais Lattices

3.2 On the Problem of Percolation

4. Conclusion and Outlook

Theoretical Background - Message Passing

Overview: Pfeilrichtungen einfügen

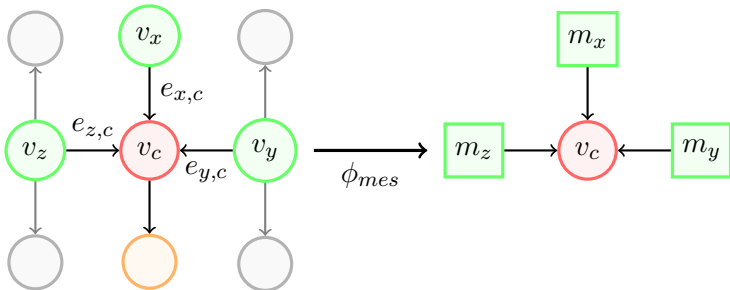


Message passing consists of three steps:

1. Computing messages (ϕ_{mes})
2. Aggregating messages (ϕ_{agg})
3. Updating node values (ϕ_{upd})

Theoretical Background - Message Passing

Step 1: Compute Messages

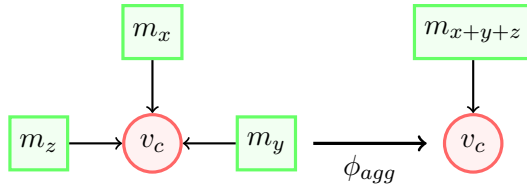


For $i \in \{x, y, z\}$ calculate:

$$m_i := \phi_{mes}(v_c, v_i, e_{i,c})$$

Theoretical Background - Message Passing

Step 2: Aggregate Messages

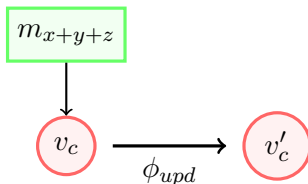


Calculate total message:

$$m_{x+y+z} := \phi_{agg}(m_x, m_y, m_z)$$

Theoretical Background - Message Passing

Step 3: Update node value



Calculate new node value:

$$v'_c := \phi_{upd}(v_c, m_{x+y+z})$$

Question: What exactly are ϕ_{mes} , ϕ_{agg} , ϕ_{upd} ?

Theoretical Background - Message Passing

Common examples for $\phi_{mes}, \phi_{agg}, \phi_{upd}$:

	GCNConv ([4])	GINEConv ([3])
$\phi_{mes}(v_c, v_i, e_{i,c})$	$W\hat{v}_i$	$\text{ReLu}(v_i + e_{i,c})$
$\phi_{agg}(m_1, \dots, m_n)$	$\sum_{i=1}^n m_i$	$\sum_{i=1}^n m_i$
$\phi_{upd}(v_c, m_{agg})$	$W\hat{v}_c + m_{agg} + b$	$h_{\theta}((1 + \epsilon)v_c + m_{agg})$

In general, $\phi_{mes}, \phi_{agg}, \phi_{upd}$ can be (almost) anything
 \implies in particular: arbitrary feed forward neural networks

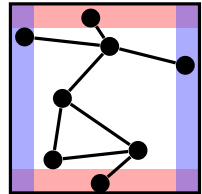
Theoretical Background - Percolation

Given are

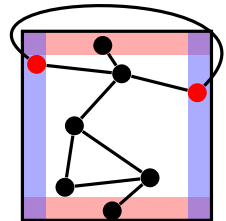
- ▶ graph $G = (V, E)$
- ▶ one position (n_x, n_y) inside unit square for each node $n \in V$
- ▶ small stripes at the edges of the unit square

G is called percolating, if there are nodes $n, m \in V$ such that

1. there is a cycle containing n and m
2. n are connected by an edge
3. n and m lie in opposite stripes



(a) non-percolating

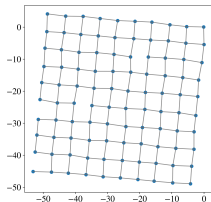


(b) percolating

Goals

Experiment 1

Input: Bravais lattice (in 2D, 3D), e.g.:



Expected output: Bravais class, e.g. square

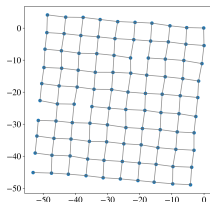
Question: What are suitable choices for ϕ 's?

Experiment 2

Goals

Experiment 1

Input: Bravais lattice (in 2D, 3D), e.g.:

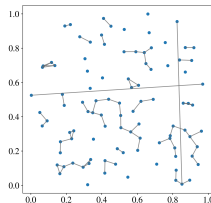


Expected output: Bravais class, e.g. square

Question: What are suitable choices for ϕ 's?

Experiment 2

Input: Graph inside unit-square, e.g.:

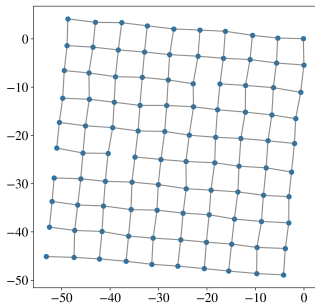


Expected output: percolating or not

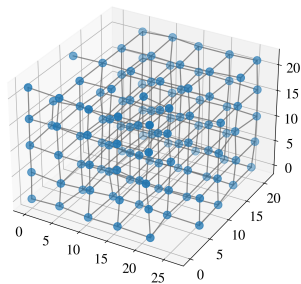
Question: Can this problem be solved by a GNN?

Results - Bravais Lattices 2D (Experiment 1)

Input data:



(a) 2d (square)



(b) 3d (cubic)

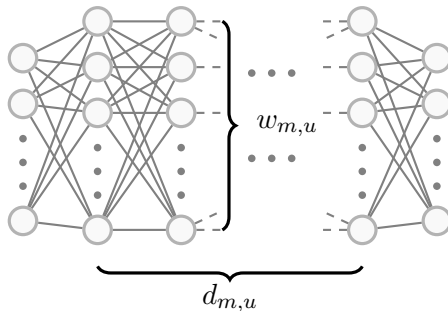
with node features = 1, edge features = direction

Expected output: Bravais class

Results - Bravais Lattices 2D (Experiment 1)

parameters varied: w_u, w_m, d_u, d_m

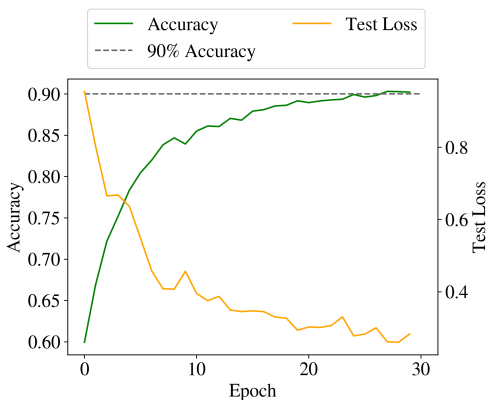
ϕ_{mes}, ϕ_{upd} = general feed forward neural network with certain width $w_{m,u}$, depth $d_{m,u}$



$$\begin{aligned} d_m &\in \{1, 2, 3\}, & w_m &\in \{10, 20, 30\}, \\ d_u &\in \{1, 2\}, & w_u &\in \{5, 10\} \end{aligned}$$

Results - Bravais Lattices 2D (Experiment 1)

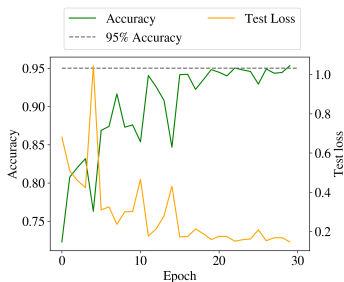
Averaged performance (over all 36 models):



Question: Do different models lead to different accuracies?

Results - Bravais Lattices 2D (Experiment 1)

Best and worst performing models:



(a) Best performing model
 $((d_m, w_m, d_u, w_u) =$
 $(2, 30, 1, 10)).$



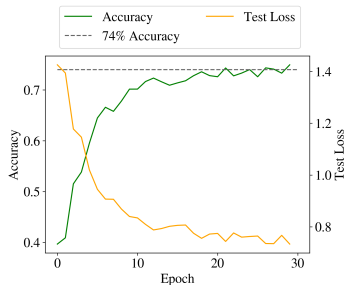
(b) Worst performing model
 $((d_m, w_m, d_u, w_u) = (3, 30, 2, 5))$

Results - Bravais Lattices 3D (Experiment 1)

Training of best and worst performing models on 3D dataset:



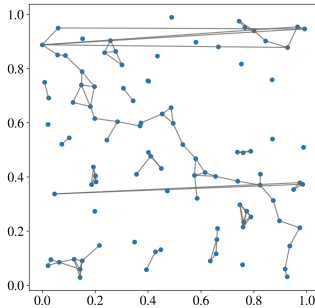
(a) Best performing model
 $((d_m, w_m, d_u, w_u) =$
 $(2, 30, 1, 10)).$



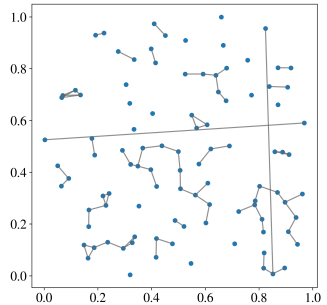
(b) Worst performing model
 $((d_m, w_m, d_u, w_u) = (3, 30, 2, 5))$

Results - Percolation (Experiment 2)

Input data:



(a) Percolating



(b) Non-Percolating

with node features = position, edge features = 1

Expected output: percolating or not

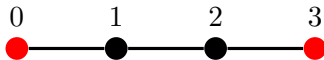
Results - Percolation (Experiment 2)

Claim: GNN can solve percolation problem \implies GNN can solve connection problem

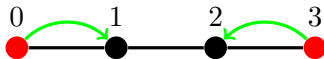
Question: Can a GNN solve the connection problem?

Answer: No!

Assumption: there is a GNN with **1** layer capable of solving the connection problem



After one message passing step:



Results - Percolation (Experiment 2)

Claim: GNN can solve percolation problem \implies GNN can solve connection problem

Question: Can a GNN solve the connection problem?

Answer: No!

Assumption: there is a GNN with 2 layers capable of solving the connection problem



After one message passing steps:



After two message passing step:



Results - Percolation (Experiment 2)

Claim: GNN can solve percolation problem \implies GNN can solve connection problem

Question: Can a GNN solve the connection problem?

Answer: No!

Assumption: there is a GNN with l layers capable of solving the connection problem

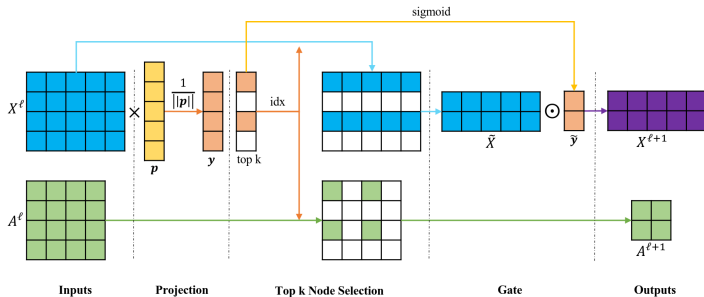


Results - Percolation (Experiment 2)

Problem: size of graph, too many nodes

⇒ pooling might help ([2])

Example: Top K Pooling ([1])



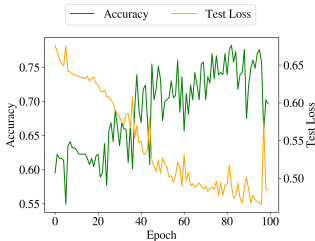
X^ℓ = node features, A^ℓ = adjacency matrix

Results - Percolation (Experiment 2)

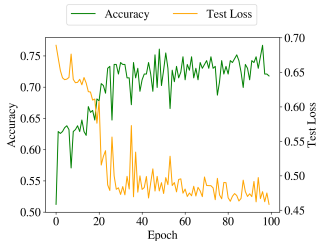
Why Top K Pooling might not help:

- ▶ deletion of nodes/edges
- ▶ connected components not preserved

Training results:



(a) Without Pooling



(b) With Pooling

Conclusion and Outlook

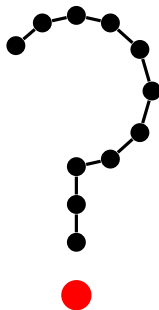
What we have learned:

- ▶ GNN capable of classifying Bravais lattices
- ▶ Correlation between width, depth and accuracies rather complicated
- ▶ Connection problem can not be solved
 \implies percolation problem can not be solved

What can be done next:

- ▶ Bravais lattices: vary node, edge features
- ▶ Percolation Problem: experiment with layers that preserve topological properties

Time for Discussion and Questions

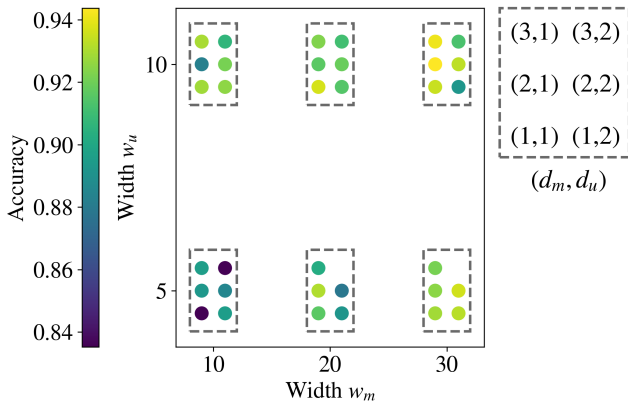


References

- [1] Hongyang Gao and Shuiwang Ji. “Graph U-Nets”. In: *CoRR* abs/1905.05178 (2019). arXiv: 1905.05178. URL: <http://arxiv.org/abs/1905.05178>.
- [2] Daniele Grattarola et al. “Understanding Pooling in Graph Neural Networks”. In: *CoRR* abs/2110.05292 (2021). arXiv: 2110.05292. URL: <https://arxiv.org/abs/2110.05292>.
- [3] Weihua Hu et al. *Strategies for Pre-training Graph Neural Networks*. 2020. arXiv: 1905.12265 [cs.LG]. URL: <https://arxiv.org/abs/1905.12265>.
- [4] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. arXiv: 1609.02907 [cs.LG]. URL: <https://arxiv.org/abs/1609.02907>.

Results - Bravais Lattices 2D (Experiment 1)

Correlation between w_m and w_u :

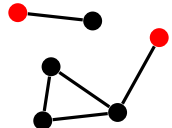
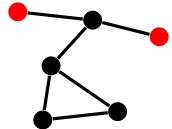


Results - Percolation (Experiment 2)

Claim: GNN can solve percolation problem \implies GNN
can solve connection problem massiv kuerzen

Procedure:

1. Start with arbitrary Graph G , select two nodes n, m

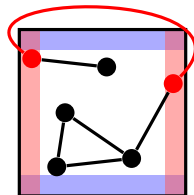
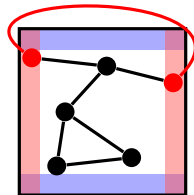


Results - Percolation (Experiment 2)

Claim: GNN can solve percolation problem \implies GNN can solve connection problem

Procedure:

1. Start with arbitrary Graph G , select two nodes n, m
2. Place G inside unit square, move n, m to edges, add edge $(n, m) \implies$ new graph \tilde{G}

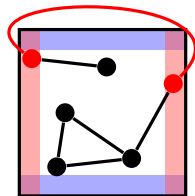
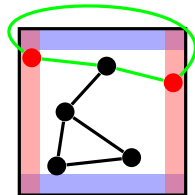


Results - Percolation (Experiment 2)

Claim: GNN can solve percolation problem \implies GNN can solve connection problem

Procedure:

1. Start with arbitrary Graph G , select two nodes n, m
2. Place G inside unit square, move n, m to edges, add edge $(n, m) \implies$ new graph \tilde{G}
3. Run GNN on $\tilde{G} \implies$ either \tilde{G} is percolating (i.e. there is a cycle containing n, m and the new edge) or not



Results - Percolation (Experiment 2)

Claim: GNN can solve percolation problem \implies GNN can solve connection problem

Procedure:

1. Start with arbitrary Graph G , select two nodes n, m
2. Place G inside unit square, move n, m to edges, add edge $(n, m) \implies$ new graph \tilde{G}
3. Run GNN on \tilde{G}
4. \tilde{G} percolating $\iff n, m$ connected in G

