Graph Neural Networks

Roger Wattenhofer

ETH Zurich – Distributed Computing Group

18th

International Conference on Distributed Computing and Intelligent Technology

Kalinga Institute of Industrial Technology

Deemed to be University, Bhubaneswar, Odisha, India

DC Track

Prof. Hagit Attiya

Technion, Israel

Prof. Philippas Tsigas

Chalmers University, Sweden

Prof. Roger Wattenhofer

ETH Zurich, Switzerland

ML Track

Prof. Matthew E. Taylor

Univ. of Alberta, Canada

Prof. Michael Cashmore

Univ. of Strathclyde, Glasgow

Prof. U. Deva Priyakumar

IIIT Hyderabad, India


DC

Graph Neural Networks

Roger Wattenhofer

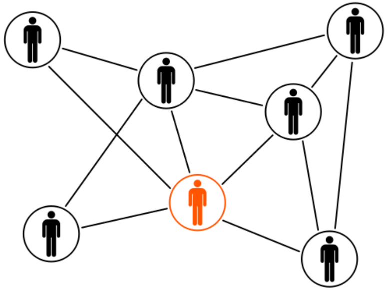
An Introduction to Graph Neural Networks from a Distributed Computing Perspective

Pál András Papp and Roger Wattenhofer

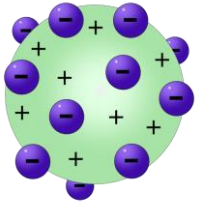
ETH Zürich, Switzerland
{apapp,wattenhofer}@ethz.ch

Abstract. The paper provides an introduction into the theoretical expressiveness of graph neural networks. We discuss the basic properties and main applications of standard GNN models, and we show how these constructions are both upper and lower bounded in expressive power by the Weisfeiler-Lehman test. We then outline a wide variety of approaches to increase the expressiveness of GNNs above this theoretical limit, and discuss the strengths and weaknesses of these methods.

social networks

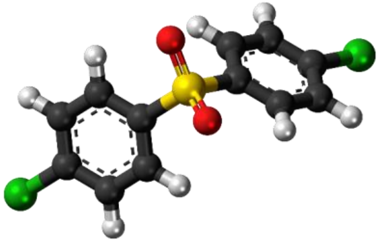


chemo-informatics



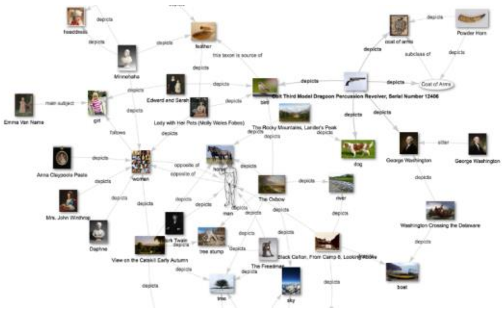
question answering systems

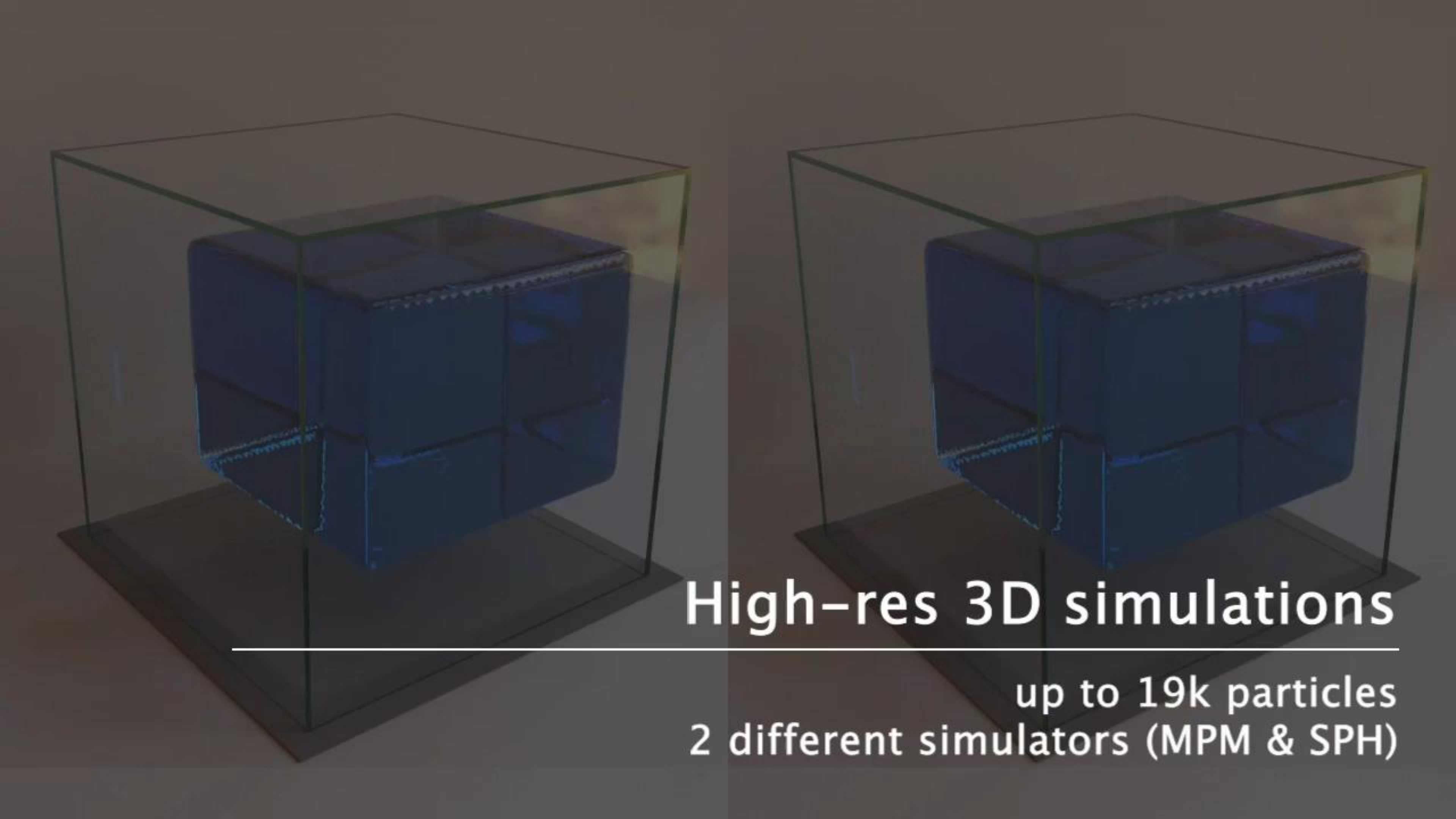
molecule recognition



recommender systems

knowledge graphs

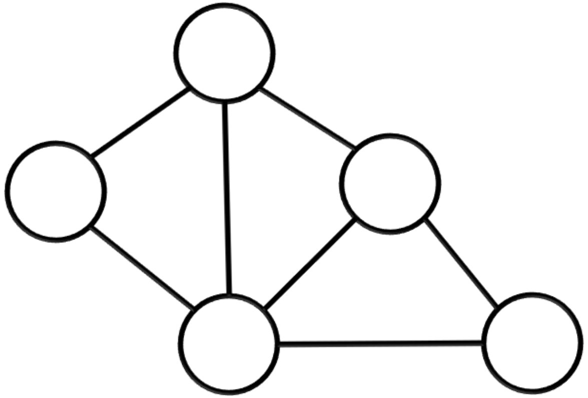



High-res 3D simulations
up to 19k particles
2 different simulators (MPM & SPH)

Distributed Computing (Message Passing)

Nodes communicate with neighbors by sending messages.

In each synchronous round, every node sends a message to its neighbors.

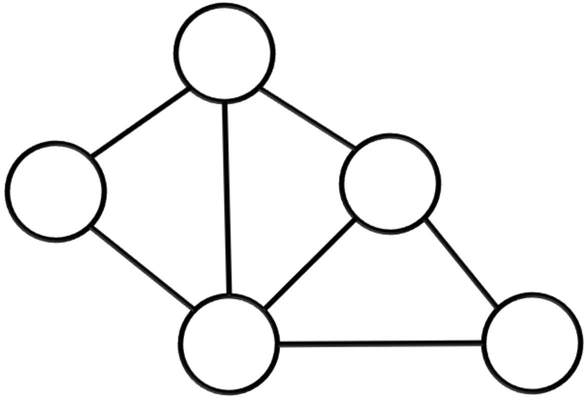


```
each round:
every node:
1. send msgs
2. rcv msgs
3. compute
```

Graph Neural Networks

Nodes communicate with neighbors by sending messages.

In each synchronous round, every node sends a message to its neighbors.



```
each round:
every node:
1. send msgs
2. rcv msgs
3. compute
```

DC Track

"Designed" algorithm

Usually node IDs

Individual messages

Solve graph problems like coloring or routing

ML Track

"Learned" algorithm

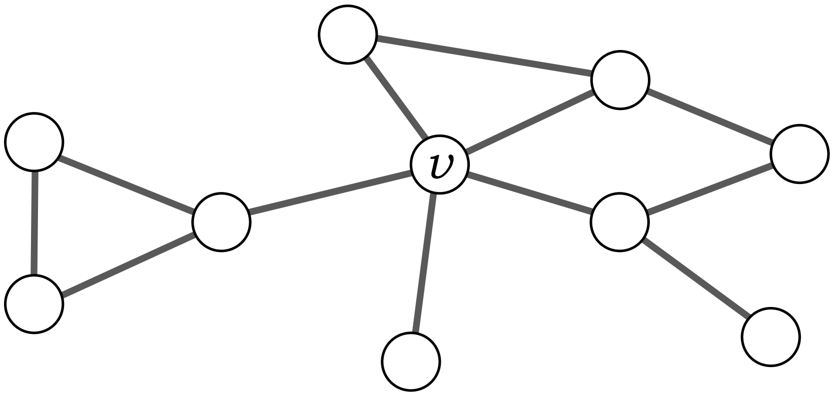
Usually node features

Aggregated messages

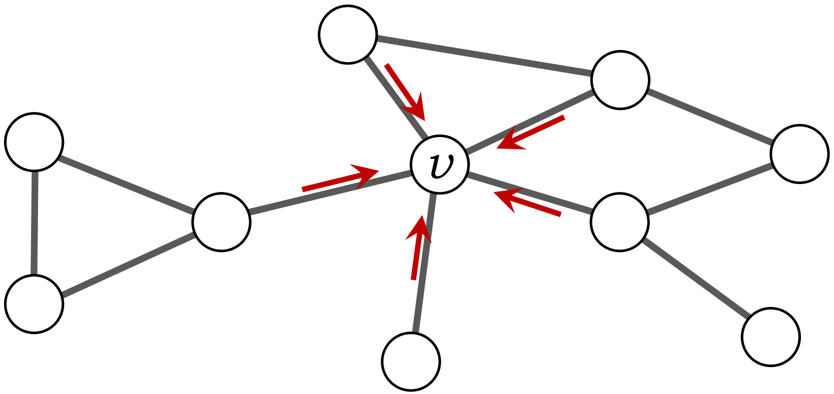
Mostly classification (node or graph)

```
each round:
every node:
1. send msgs
2. rcv msgs
3. compute
```

Message Passing GNNs

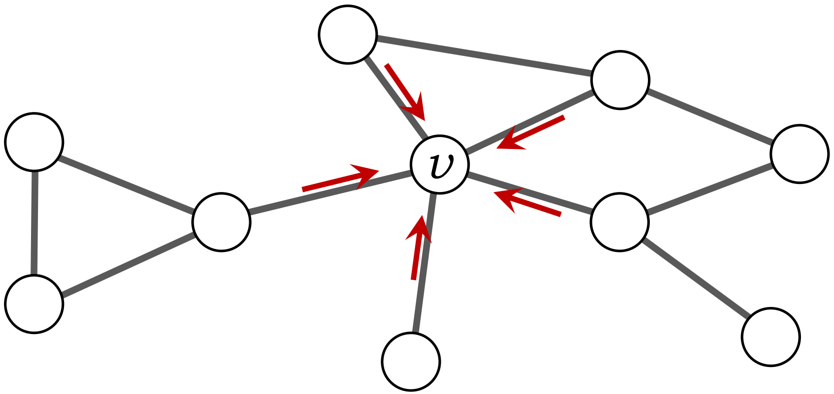


Message Passing GNNs



$$a_v = \textsc{Aggregate}\left(\{\{\, h_u \mid u \in N(v) \,\}\}\right)$$

Message Passing GNNs

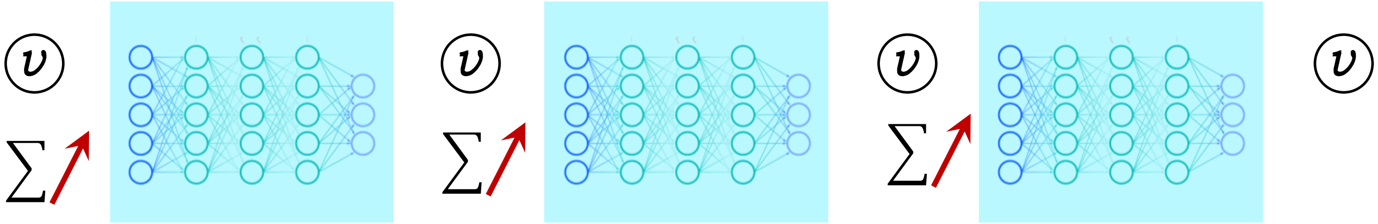


$$a_v = \textsc{Aggregate}\,(\,\{\{\, h_u \mid u \in N(v) \,\}\}\,)$$

$$h_v^{(t+1)} = \textsc{Update}\,(\, h_v,\, a_v \,)$$

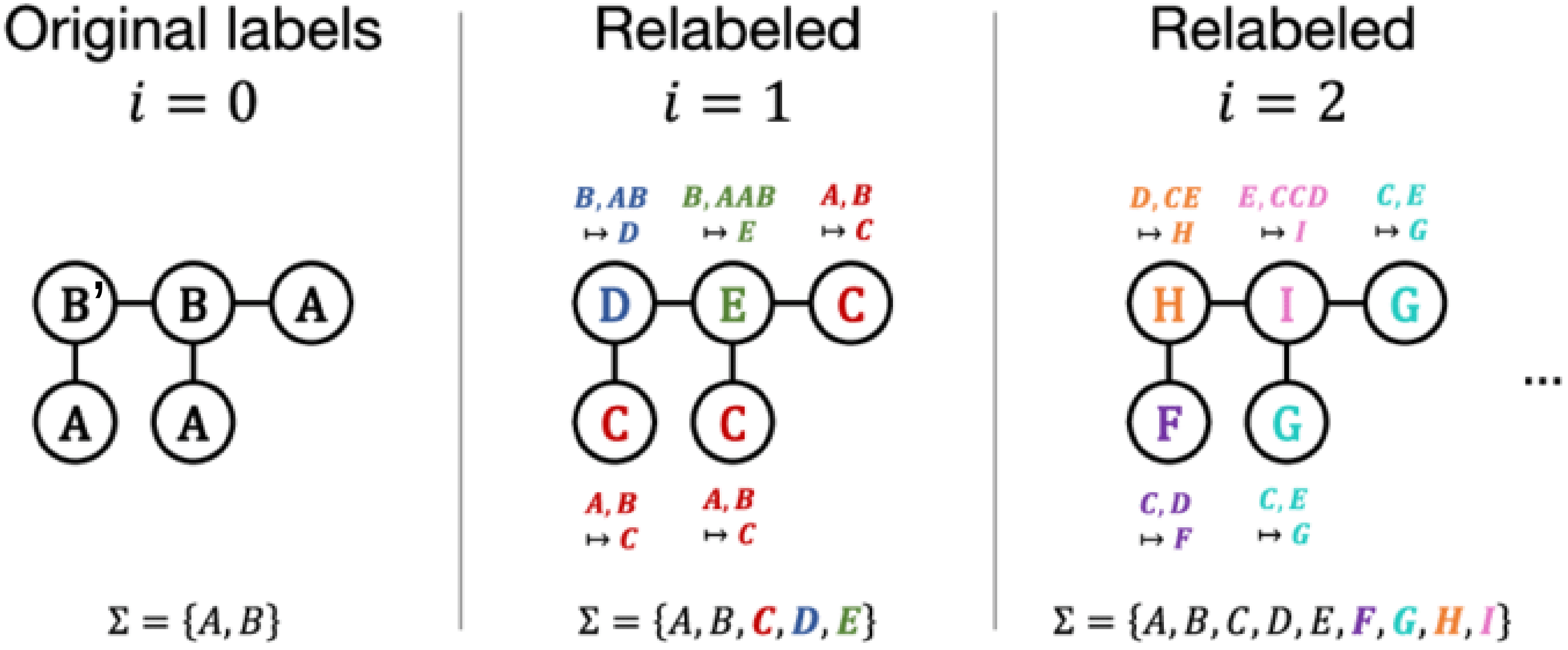
Message Passing GNNs

Message Passing GNNs

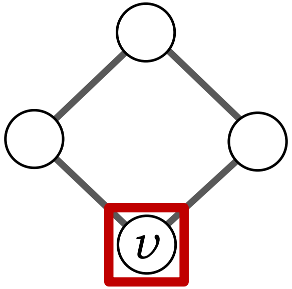


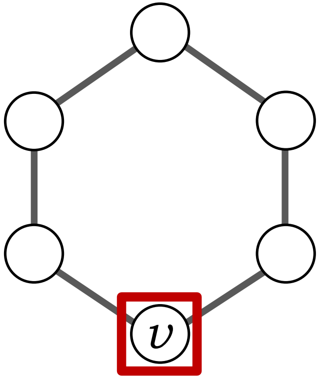
Limitations of GNNs?

Weisfeiler-Lehman Graph Isomorphism Test

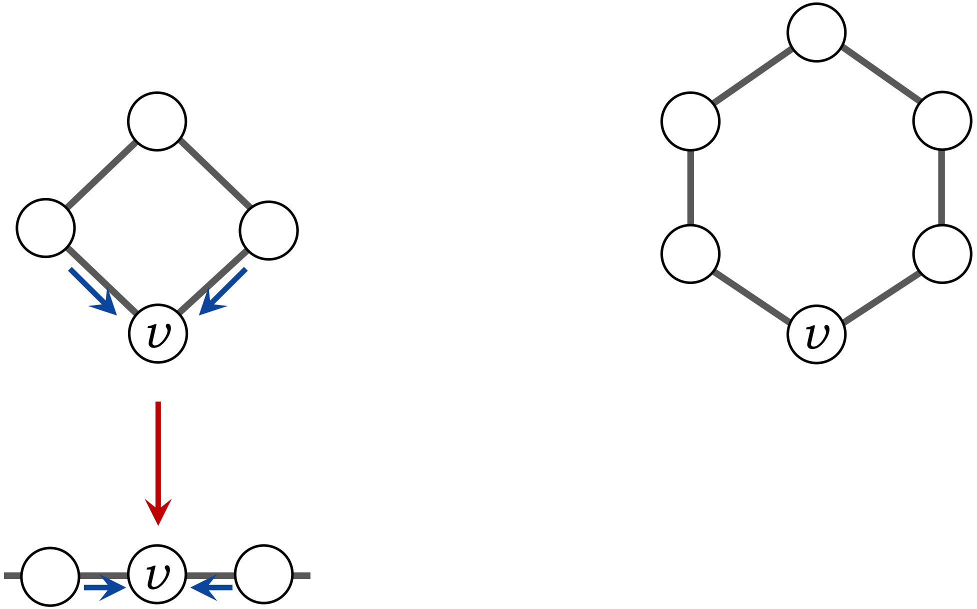


Limits of GNNs

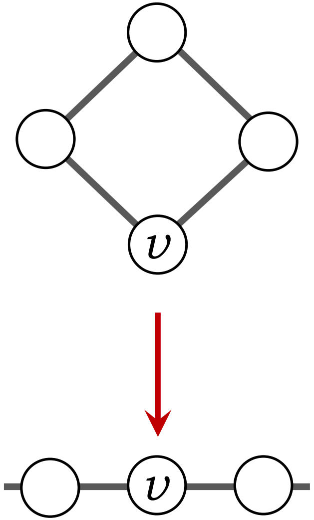


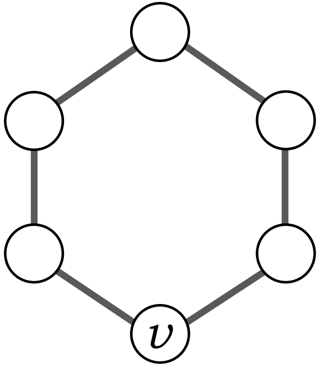


Limits of GNNs

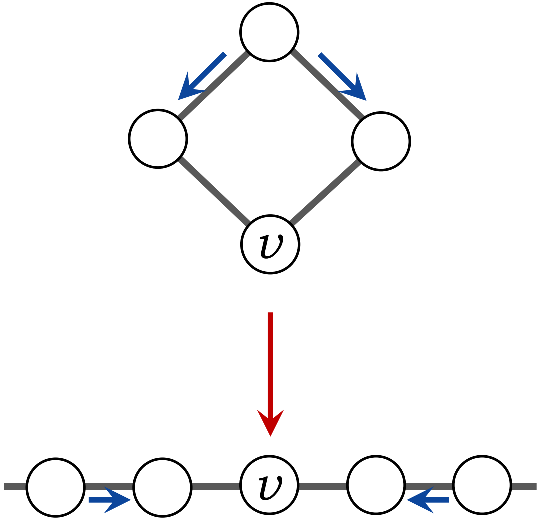


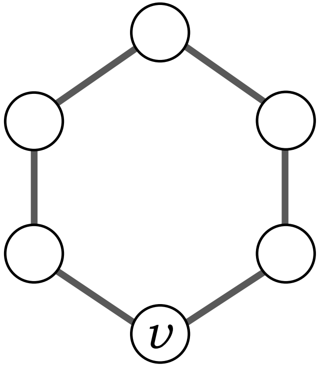
Limits of GNNs



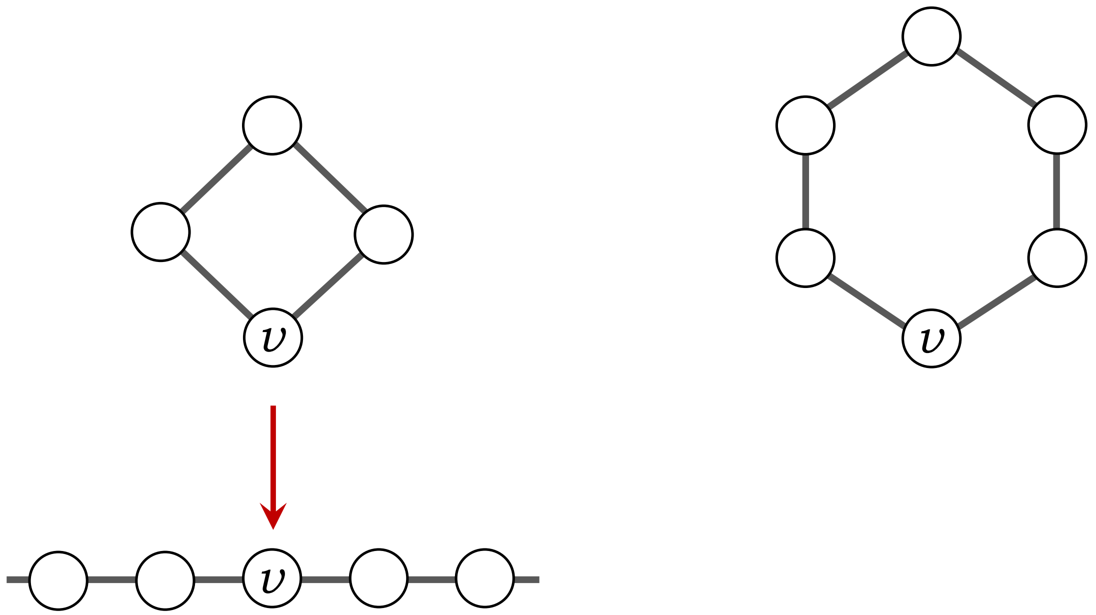


Limits of GNNs

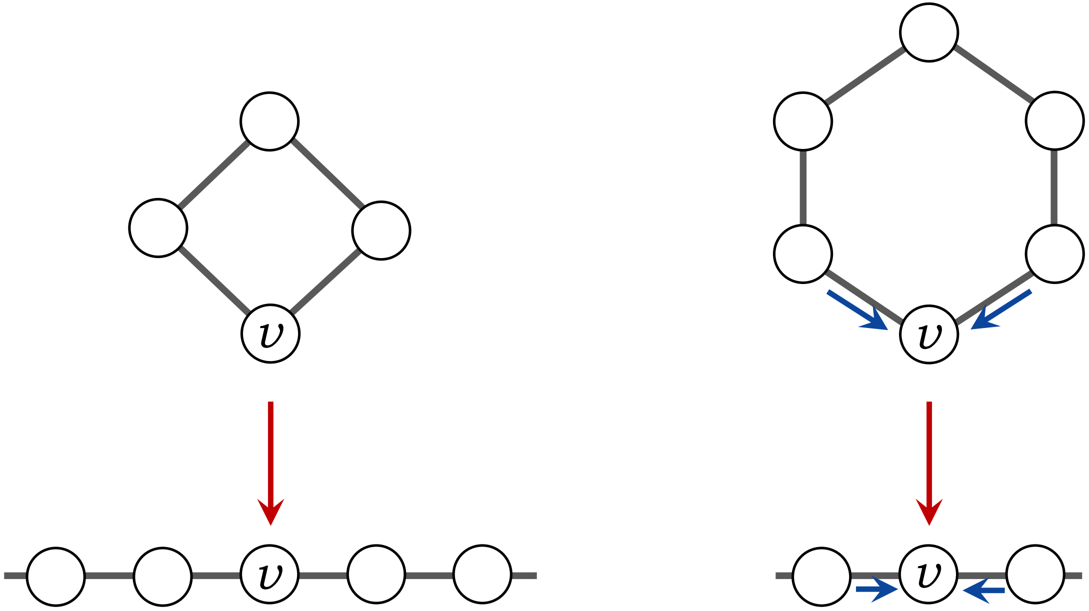




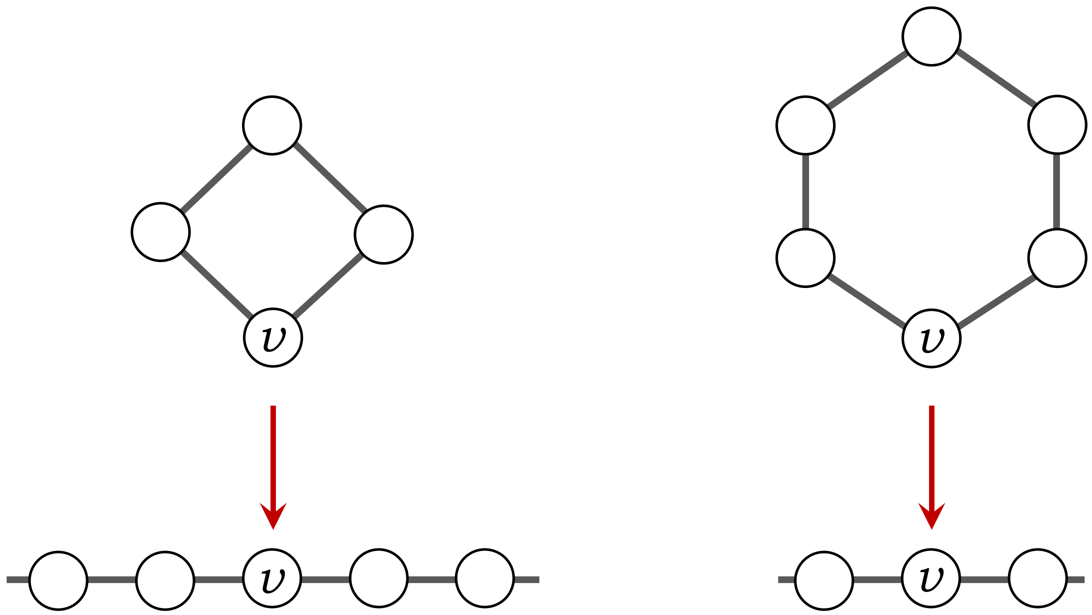
Limits of GNNs



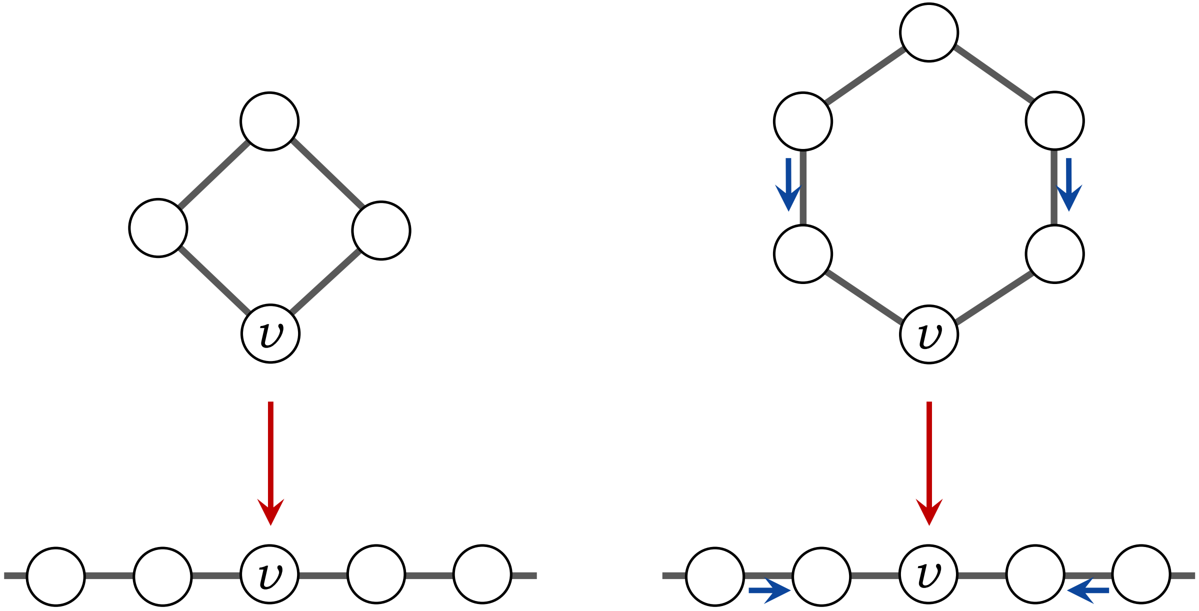
Limits of GNNs



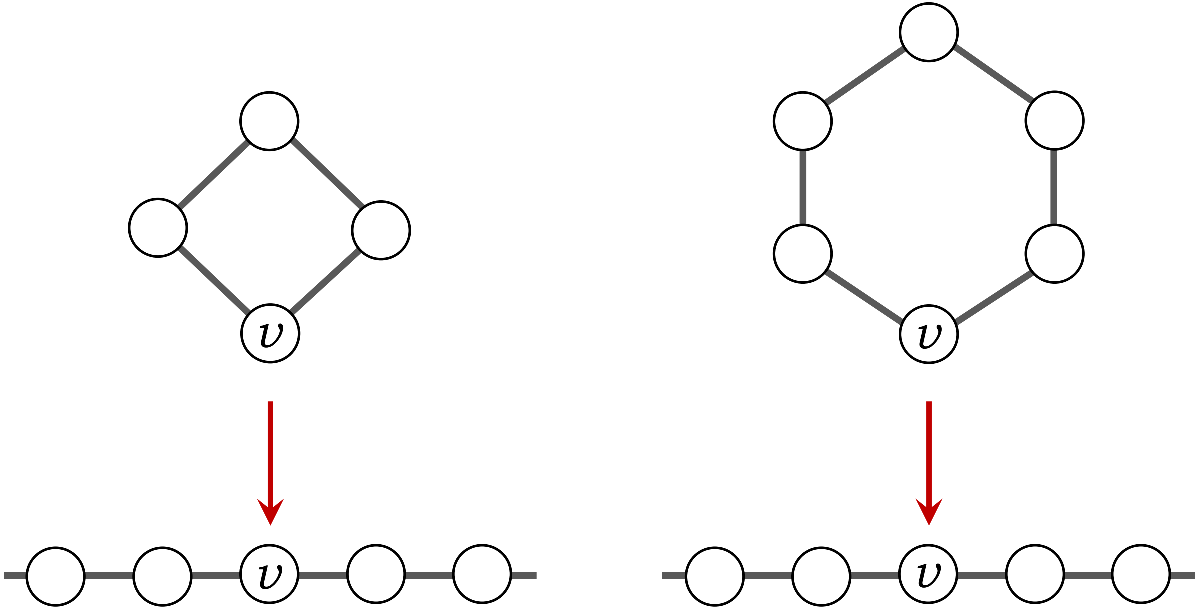
Limits of GNNs



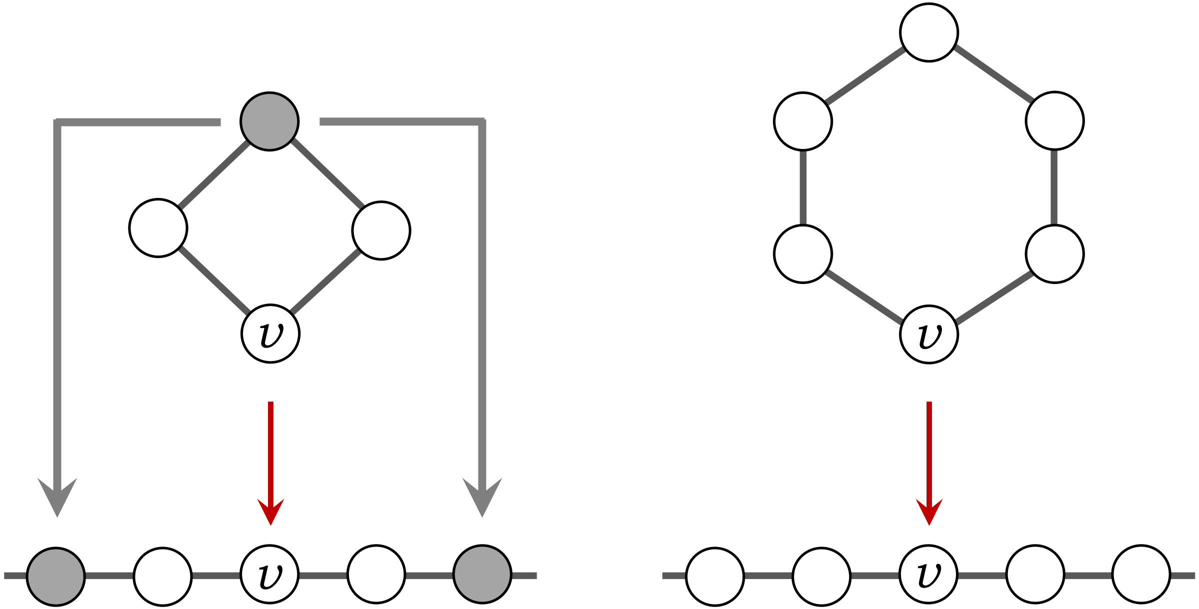
Limits of GNNs



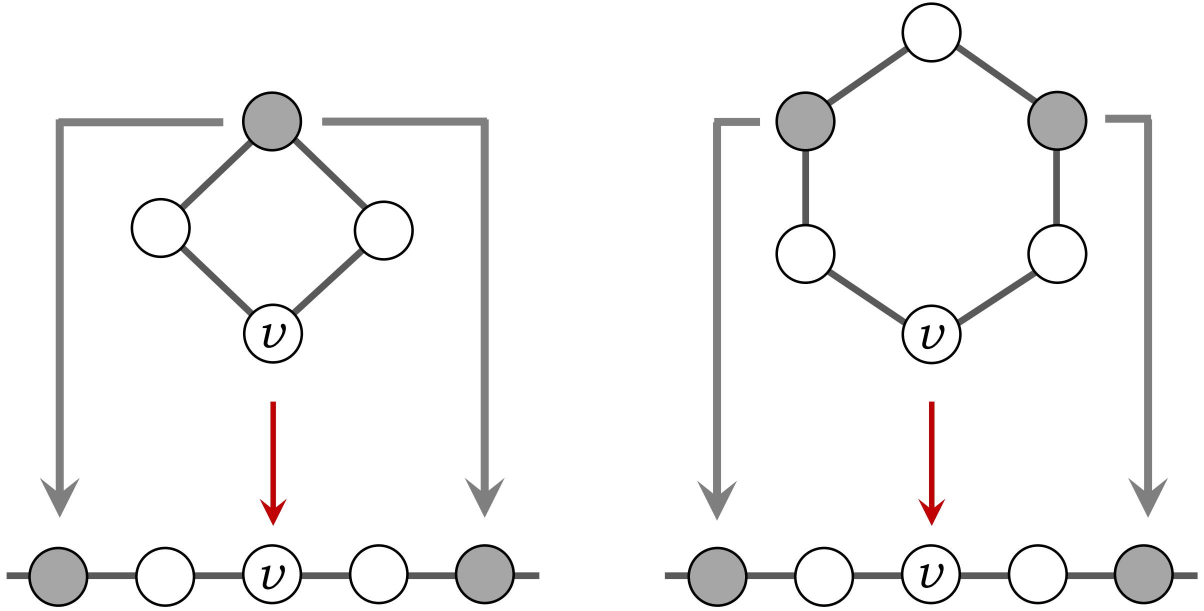
Limits of GNNs



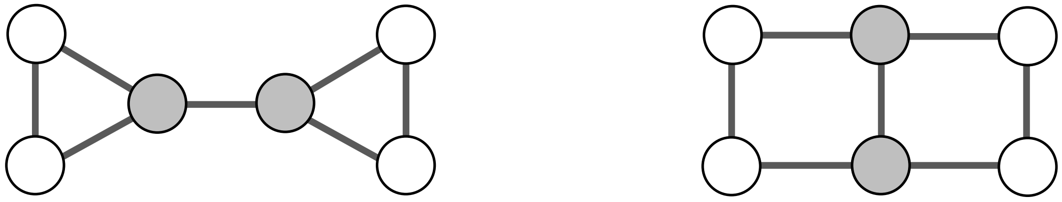
Limits of GNNs



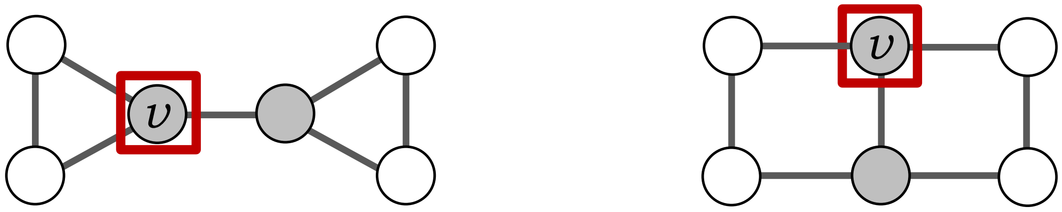
Limits of GNNs



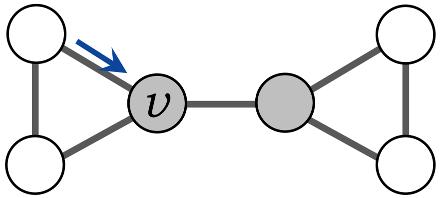
Limits of GNNs

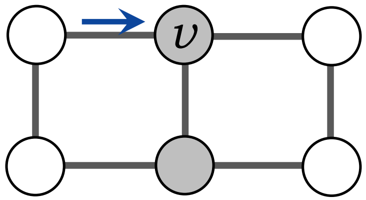


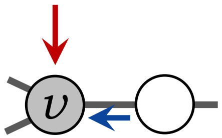
Limits of GNNs

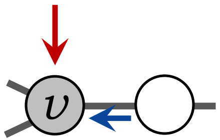


Limits of GNNs

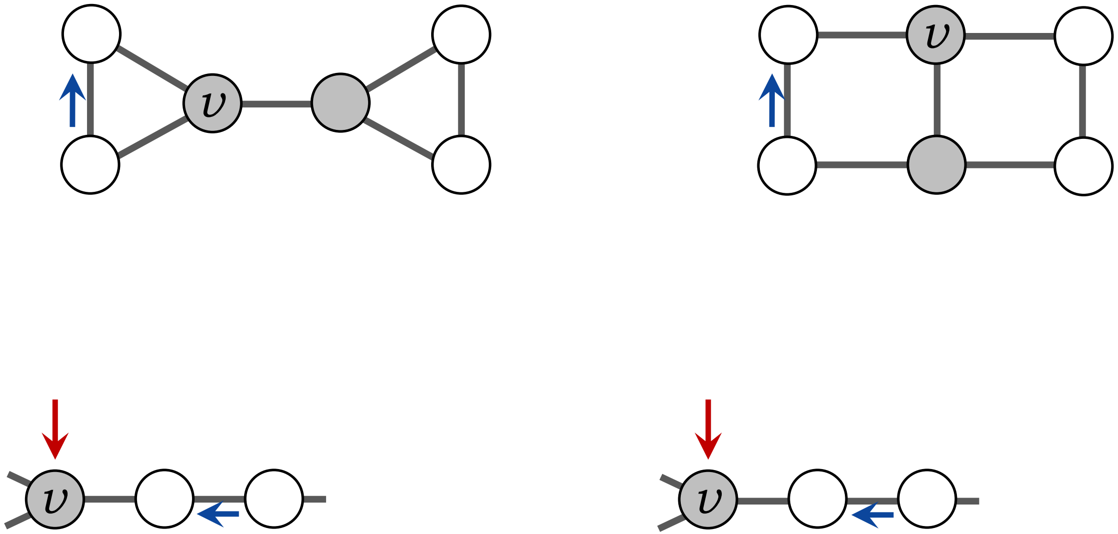




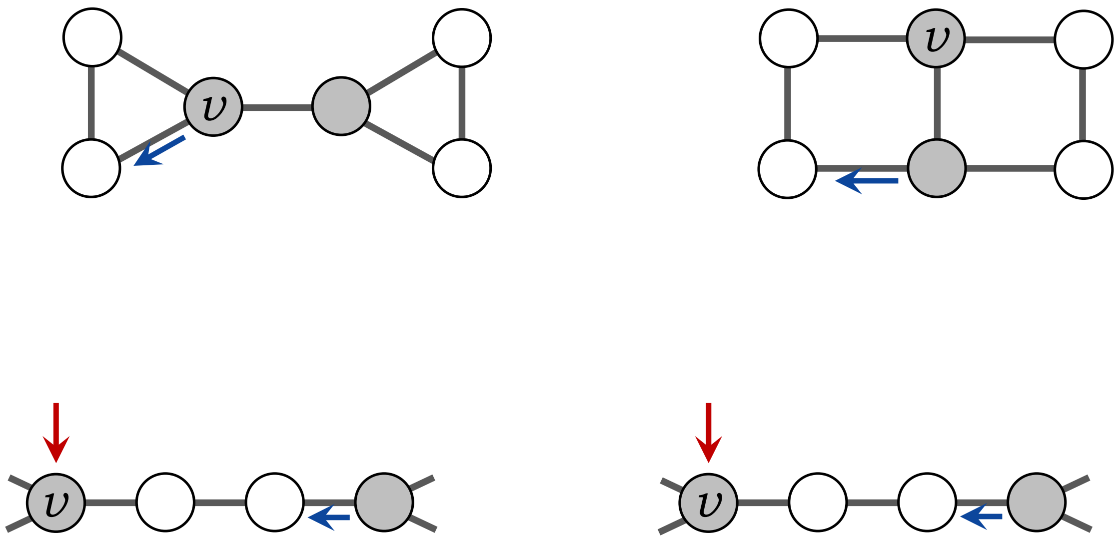




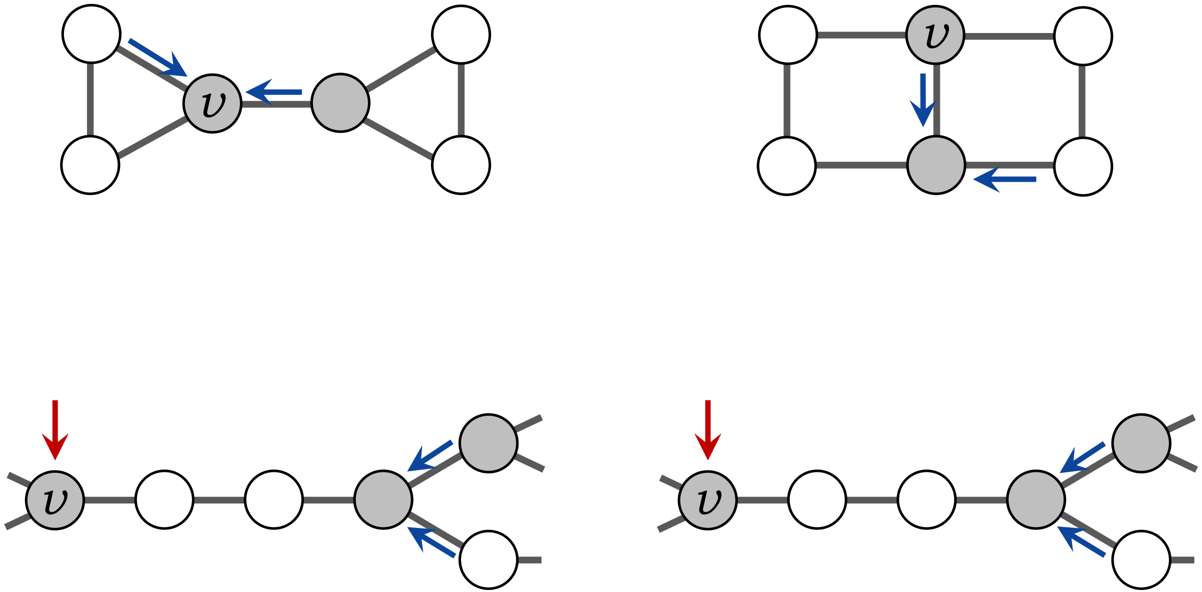
Limits of GNNs



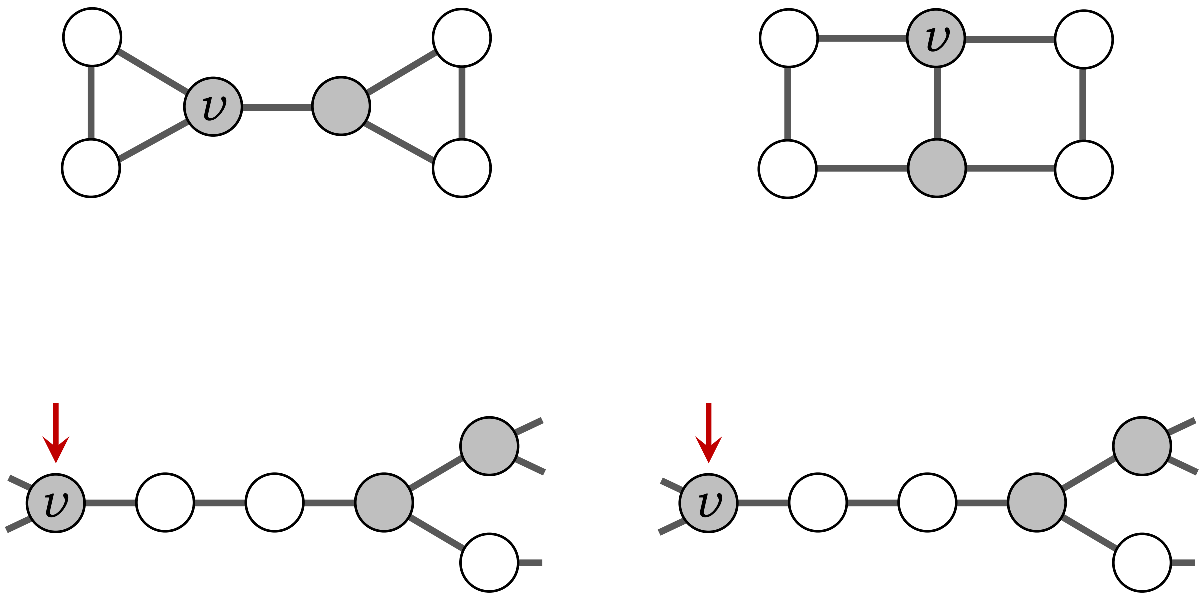
Limits of GNNs



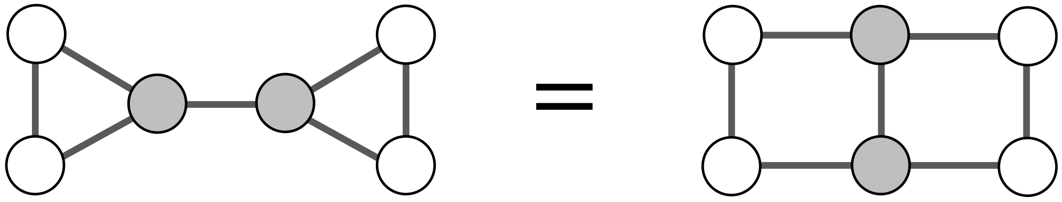
Limits of GNNs



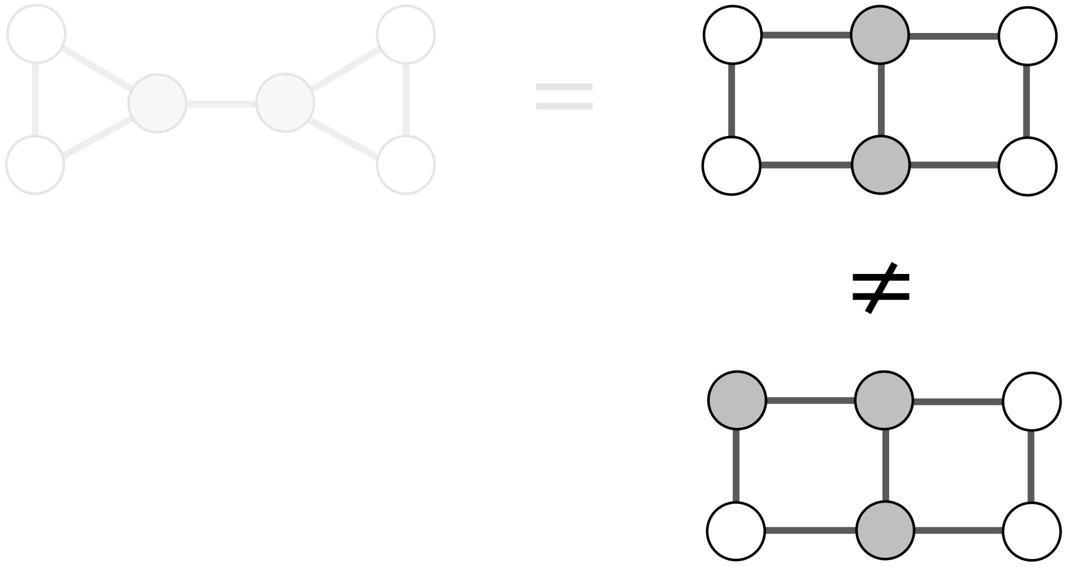
Limits of GNNs



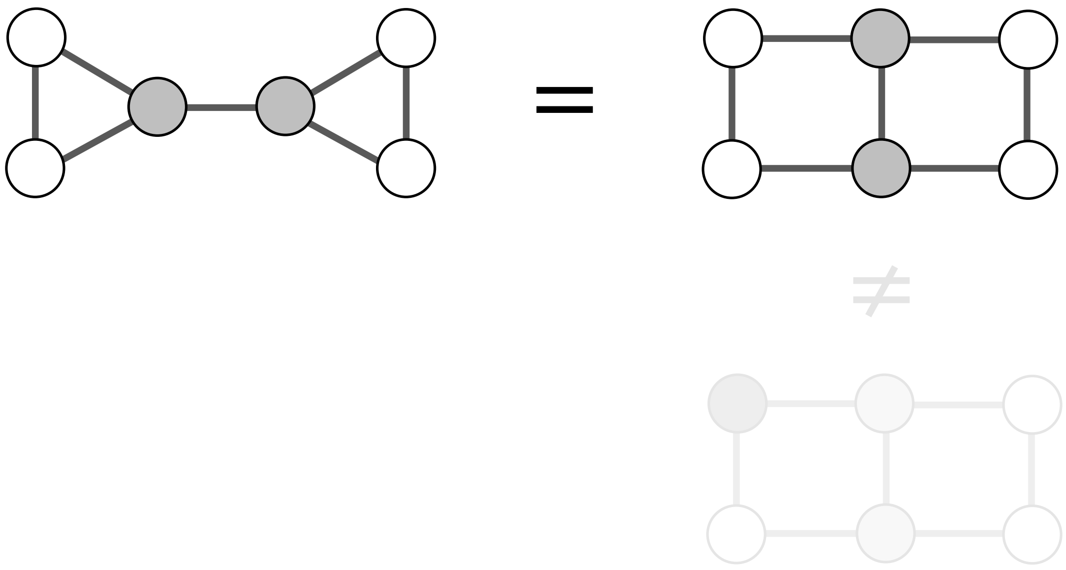
Graph Neural Networks



Graph Neural Networks



Graph Neural Networks



More Expressive GNNs?

→ run GNN on metagraph

→ extend GNN model

→ add random features

→ ***DropGNN:*** *GNNs with dropouts*

DropGNN: Random Dropouts Increase the Expressiveness of Graph Neural Networks

Pál András Papp
ETH Zurich
apapp@ethz.ch

Karolis Martinkus
ETH Zurich
martinkus@ethz.ch

Lukas Faber
ETH Zurich
lfaber@ethz.ch

Roger Wattenhofer
ETH Zurich
wattenhofer@ethz.ch

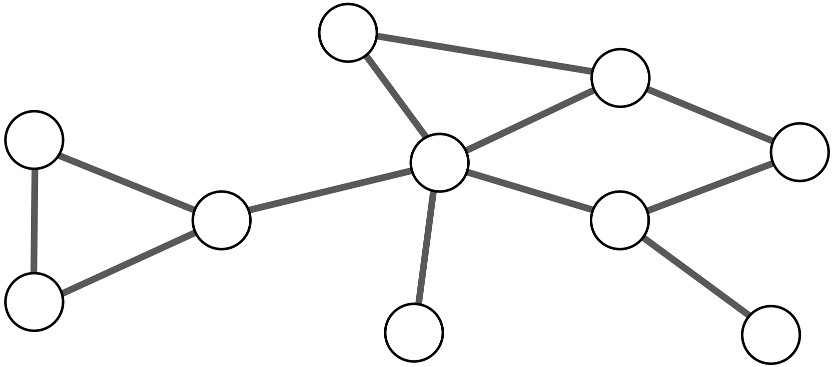
Abstract

This paper studies Dropout Graph Neural Networks (DropGNNs), a new approach that aims to overcome the limitations of standard GNN frameworks. In DropGNNs, we execute multiple runs of a GNN on the input graph, with some of the nodes randomly and independently dropped in each of these runs. Then, we combine the results of these runs to obtain the final result. We prove that DropGNNs can distinguish various graph neighborhoods that cannot be separated by message passing GNNs. We derive theoretical bounds for the number of runs required to ensure a reliable distribution of dropouts, and we prove several properties regarding the expressive capabilities and limits of DropGNNs. We experimentally validate our theoretical findings on expressiveness. Furthermore, we show that DropGNNs perform competitively on established GNN benchmarks.

GNNs with Dropouts

Multiple runs of the GNN

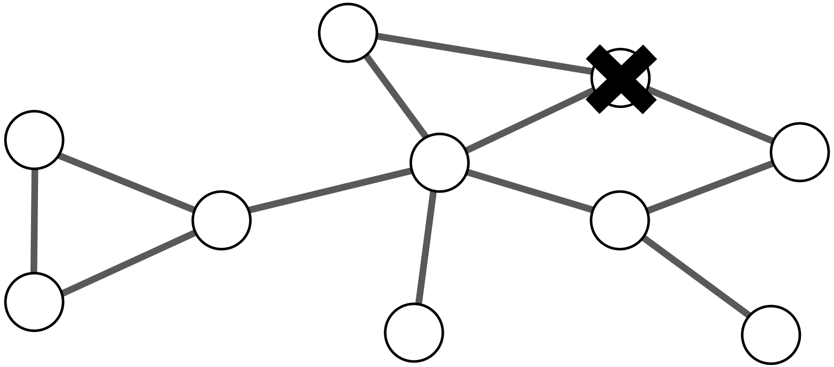
Each node removed with probability *p* independently



GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

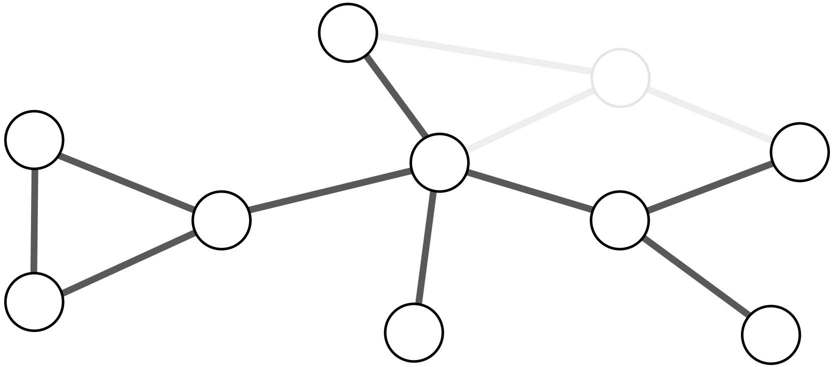


Run #1

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

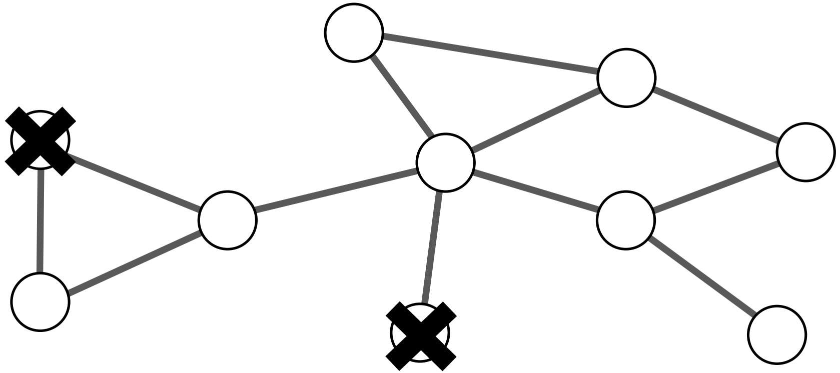


Run #1

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

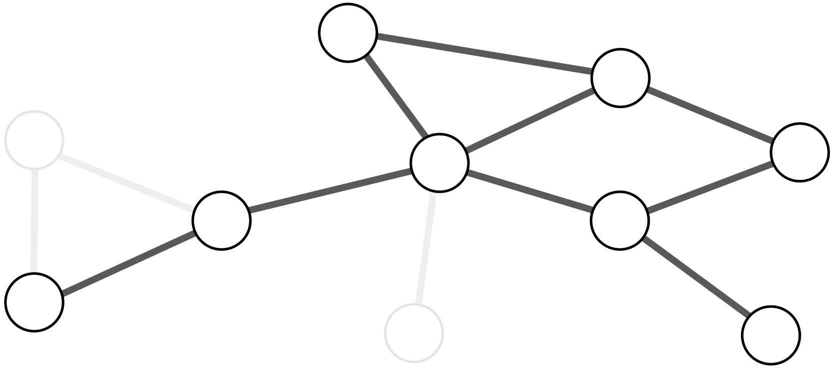


Run #2

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

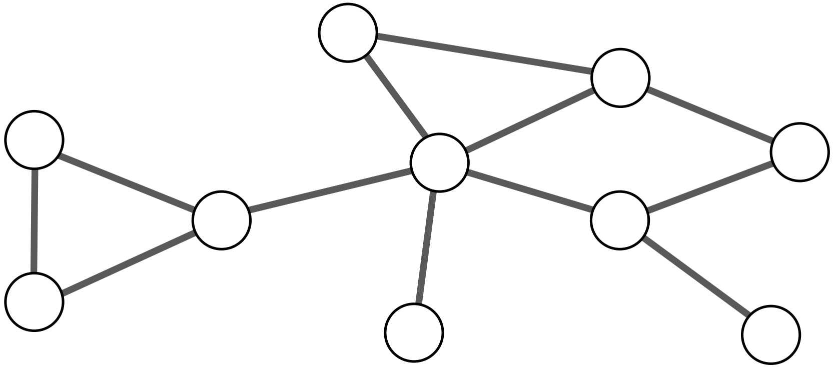


Run #2

GNNs with Dropouts

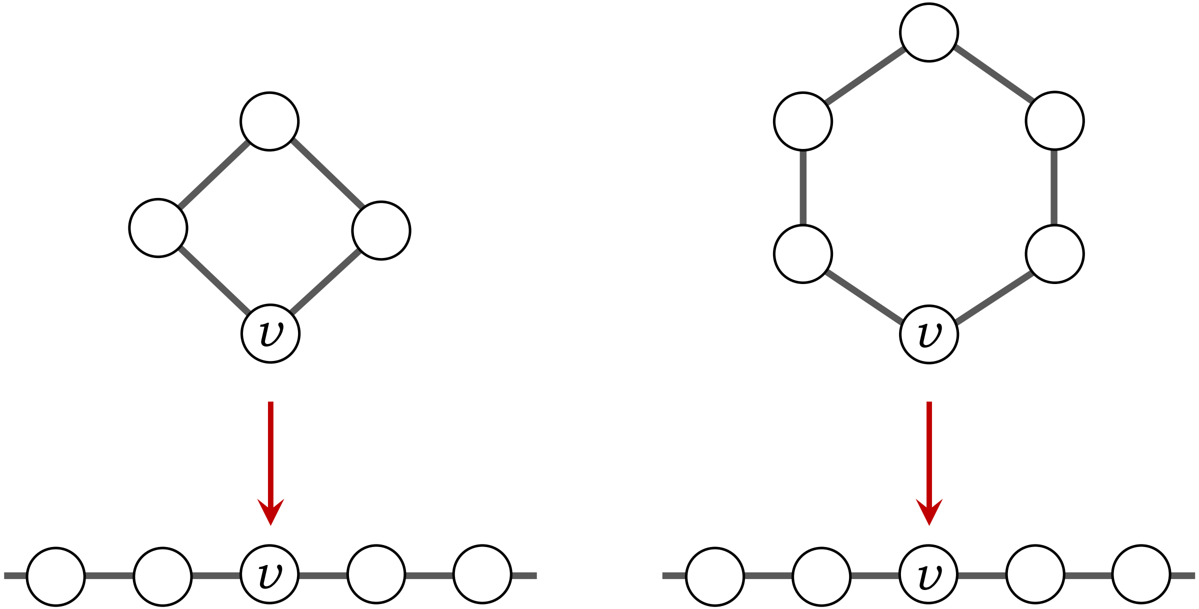
Multiple runs of the GNN

Each node removed with probability *p* independently

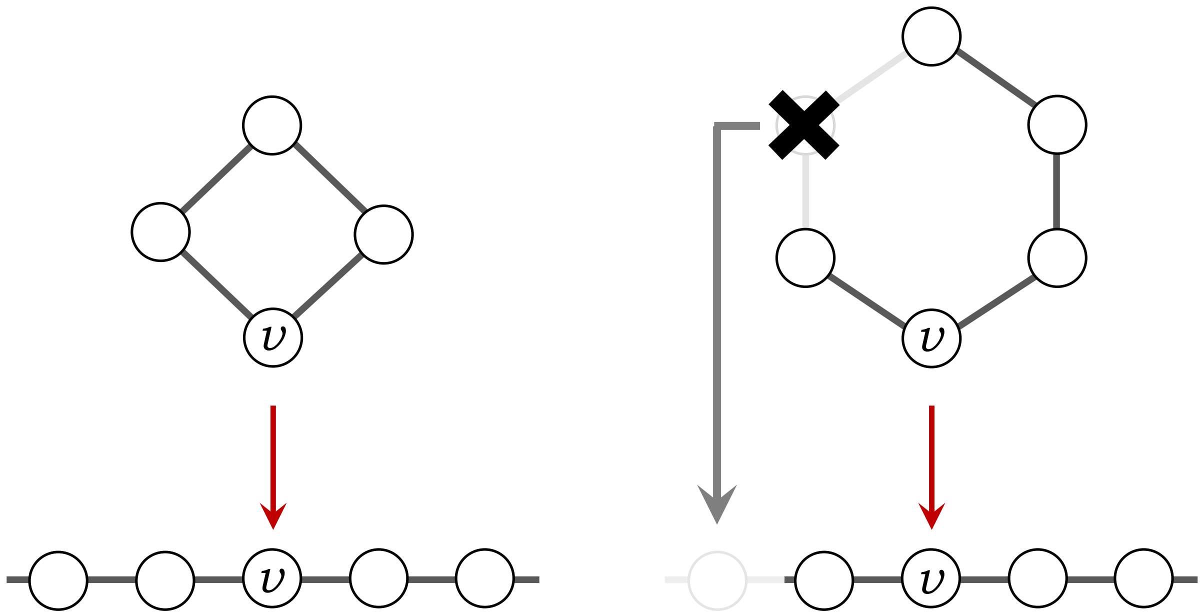


Run #3

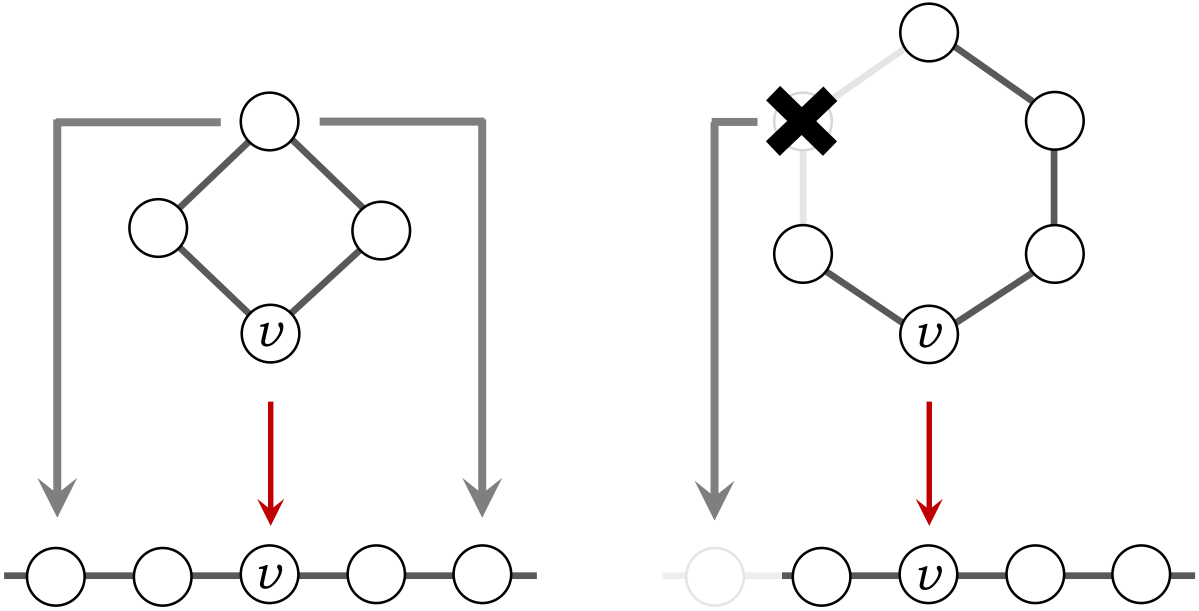
GNNs with Dropouts



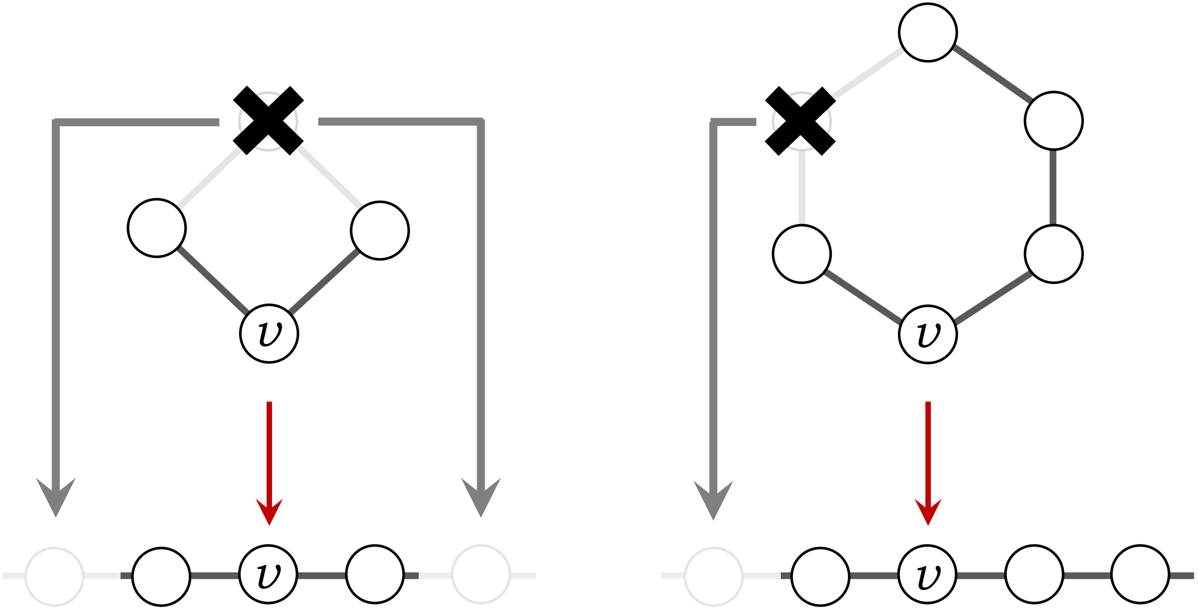
GNNs with Dropouts



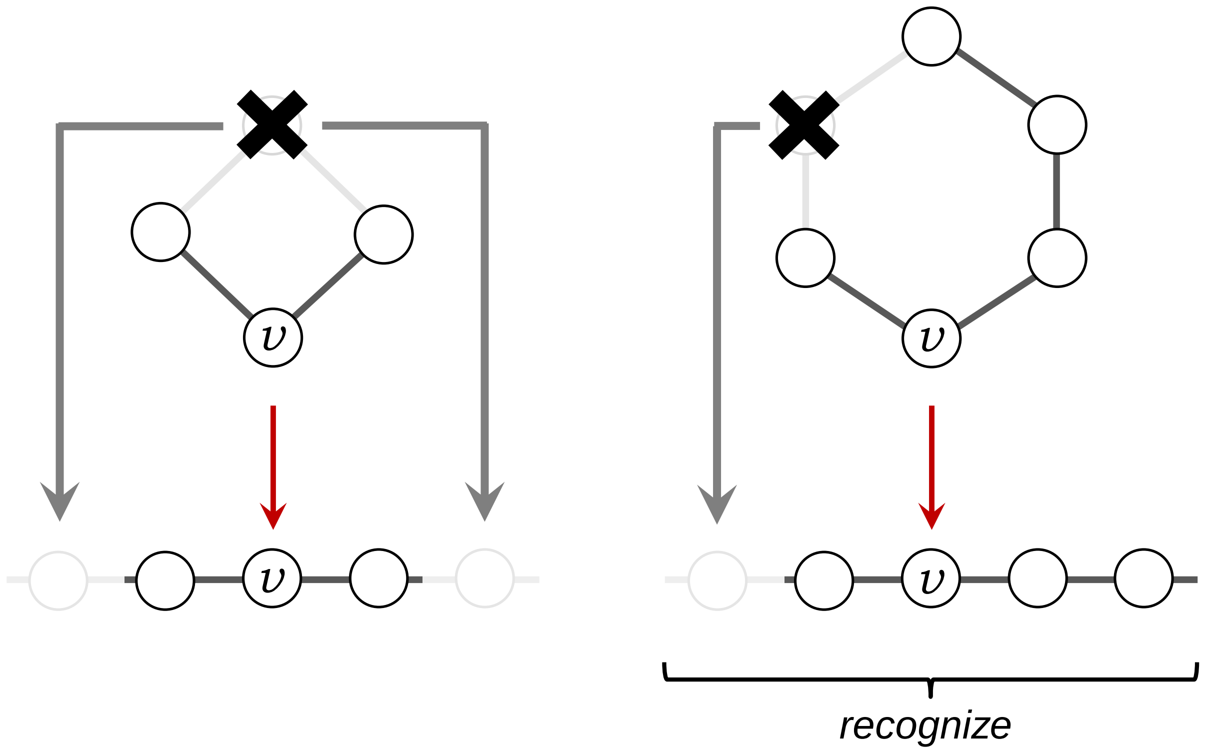
GNNs with Dropouts



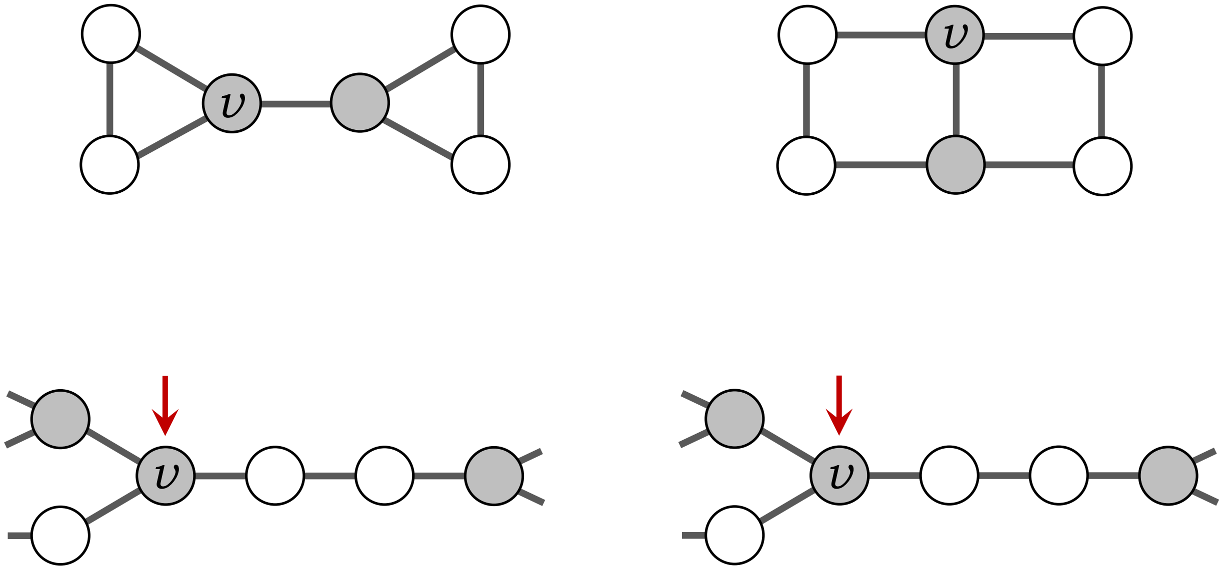
GNNs with Dropouts



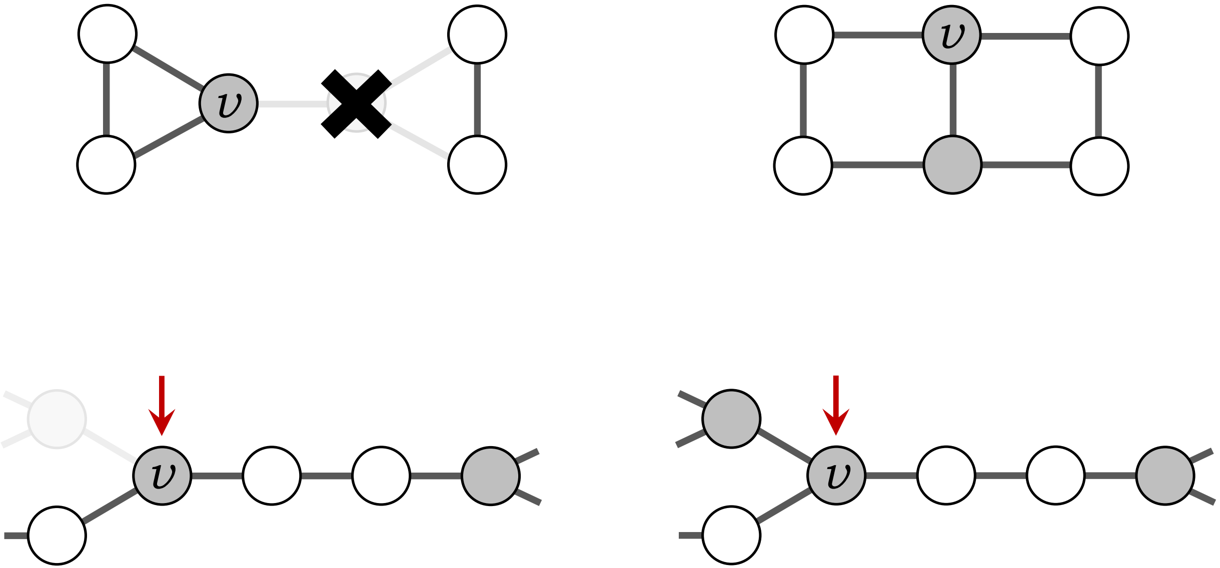
GNNs with Dropouts



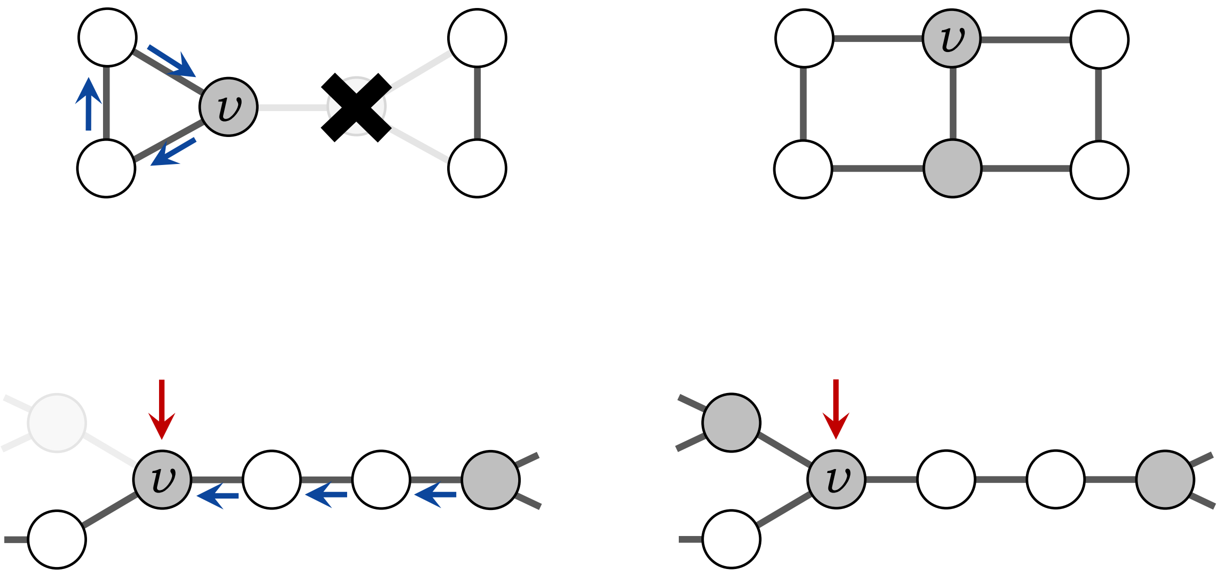
GNNs with Dropouts



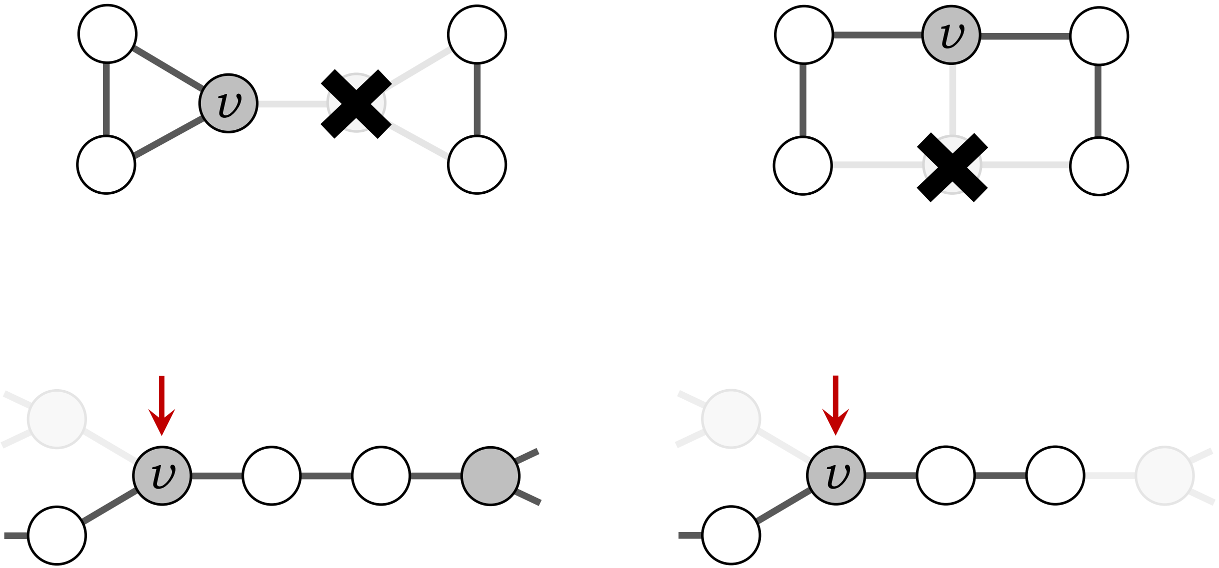
GNNs with Dropouts



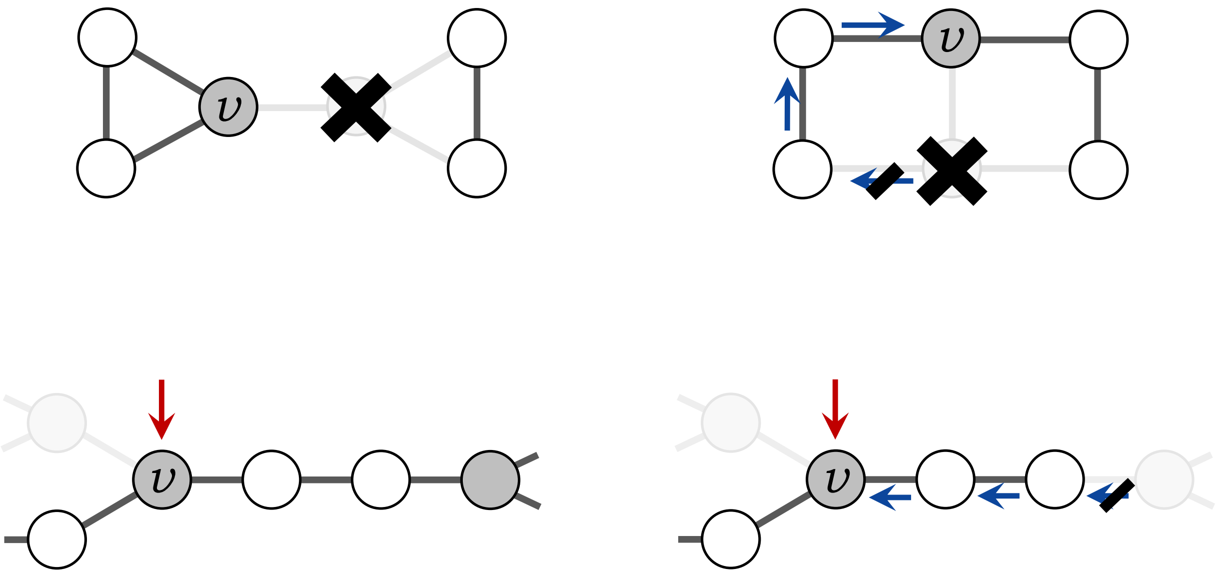
GNNs with Dropouts



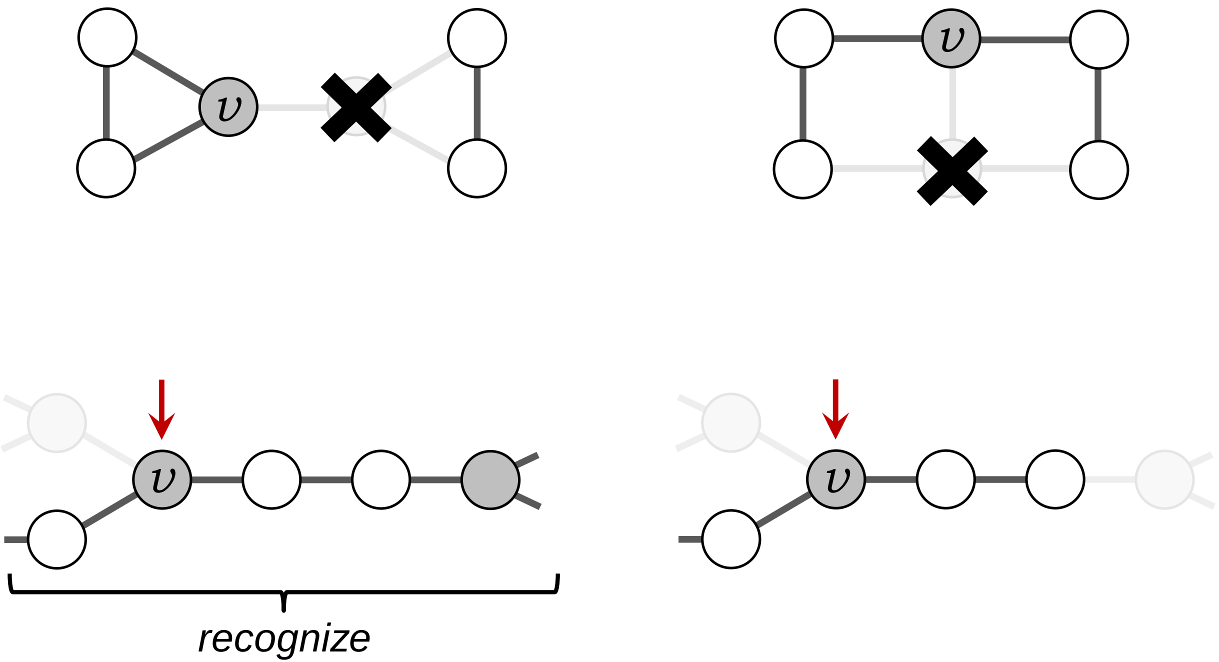
GNNs with Dropouts



GNNs with Dropouts



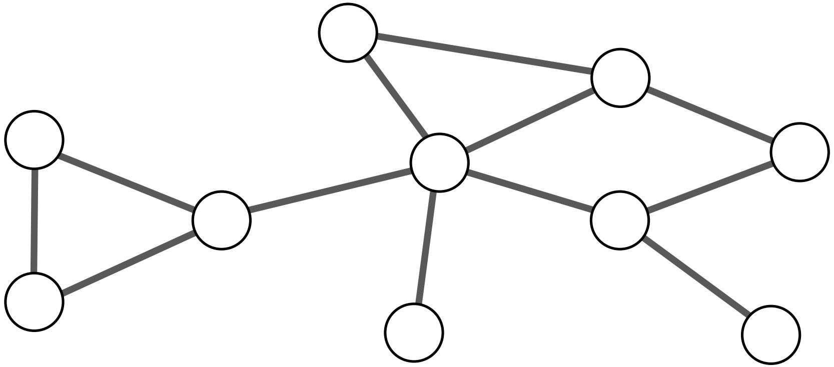
GNNs with Dropouts



GNNs with Dropouts

Multiple runs of the GNN

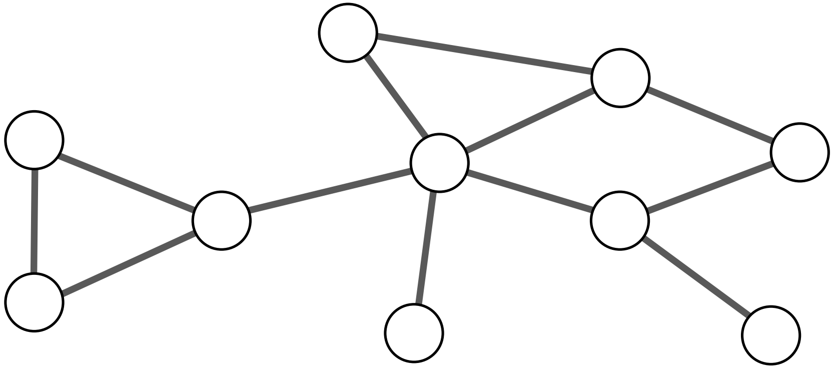
Each node removed with probability *p* independently



GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

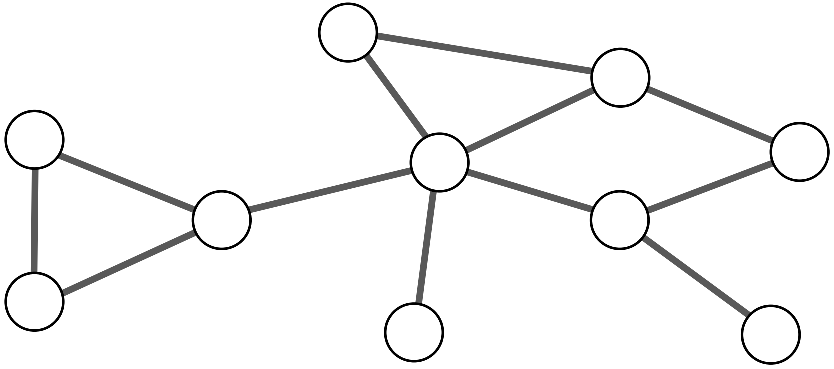


$$h_v = \textsc{RunAggregate}\,(h_v^{[1]}, h_v^{[2]}, \ldots, h_v^{[r]})$$

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

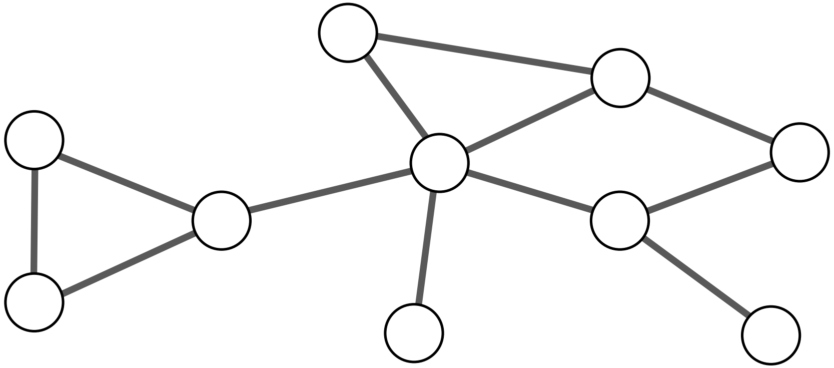


$$h_v = \textsc{RunAggregate}\left(\boxed{h_v^{[1]}}, h_v^{[2]}, \ldots, h_v^{[r]}\right)$$

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

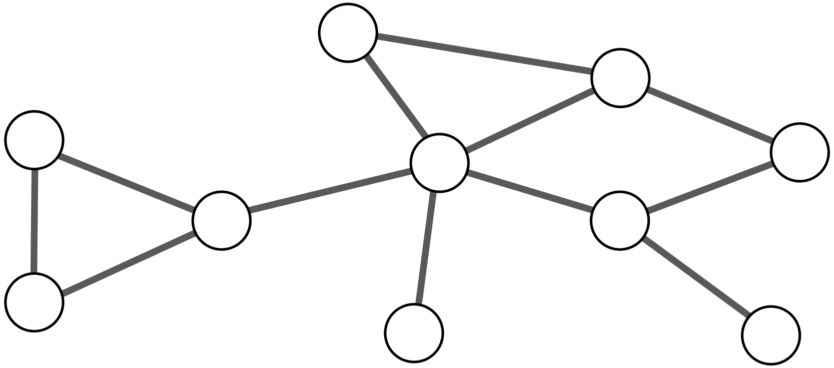


$$h_v = \textsc{RunAggregate}\left(h_v^{[1]}, \boxed{h_v^{[2]}}, \ldots, h_v^{[r]}\right)$$

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently



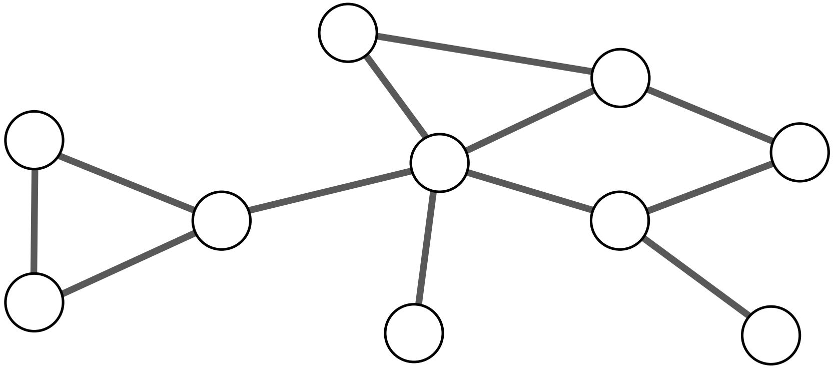
$$h_\upsilon = \textsc{RunAggregate}\,(h_\upsilon^{[1]}, h_\upsilon^{[2]}, \ldots, h_\upsilon^{[r]})$$

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

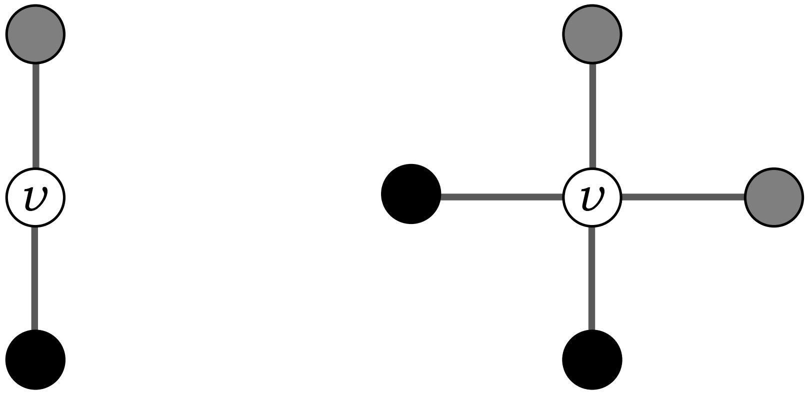
both training and testing!



$$h_v = \textsc{RunAggregate}\,(h_v^{[1]}, h_v^{[2]}, \dots , h_v^{[r]})$$

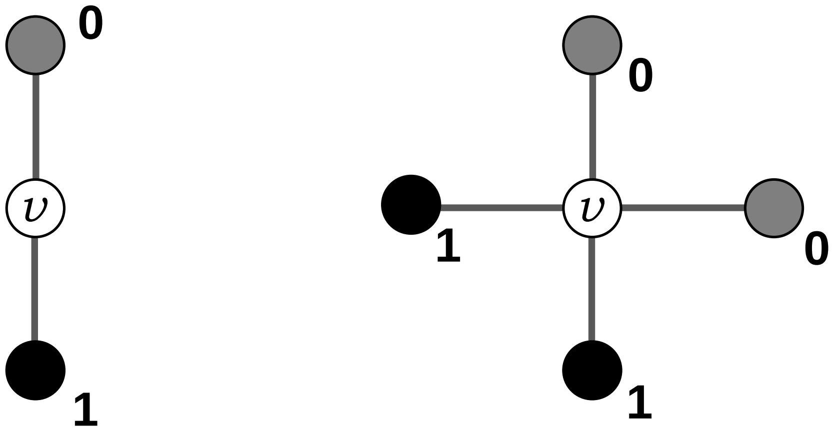
GNNs with Dropouts

MEAN aggregation of neighbors



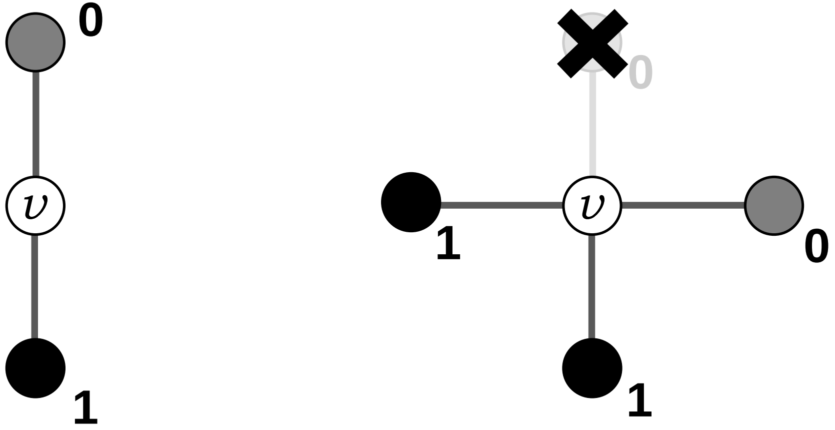
GNNs with Dropouts

MEAN aggregation of neighbors



GNNs with Dropouts

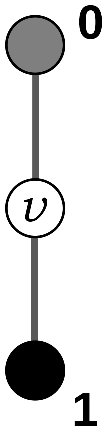
Mean aggregation of neighbors



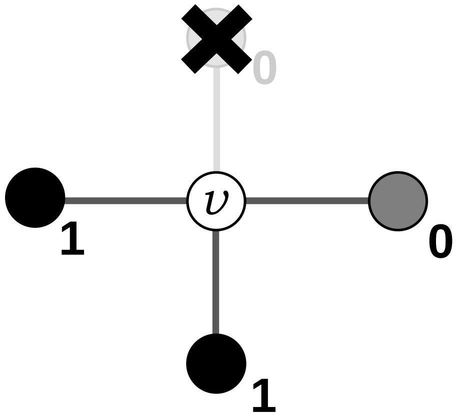
Mean = 0.66

GNNs with Dropouts

MEAN aggregation of neighbors



MEAN $\in \{0, 0.5, 1\}$



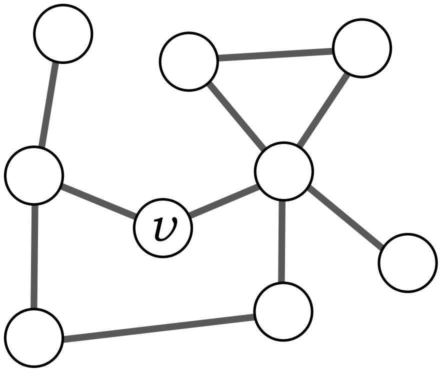
MEAN = 0.66

DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

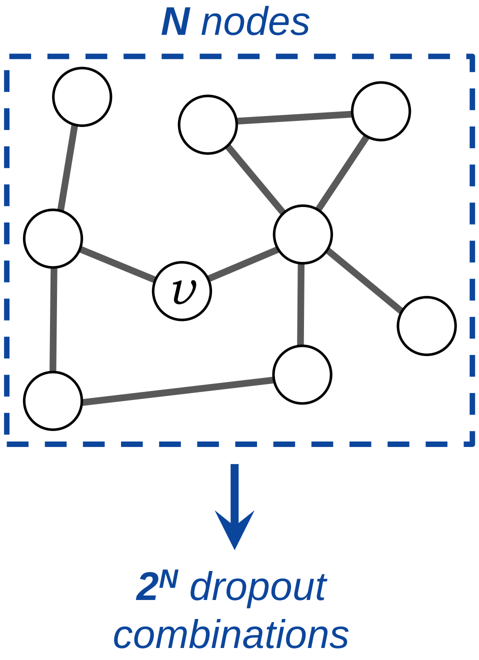


DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead



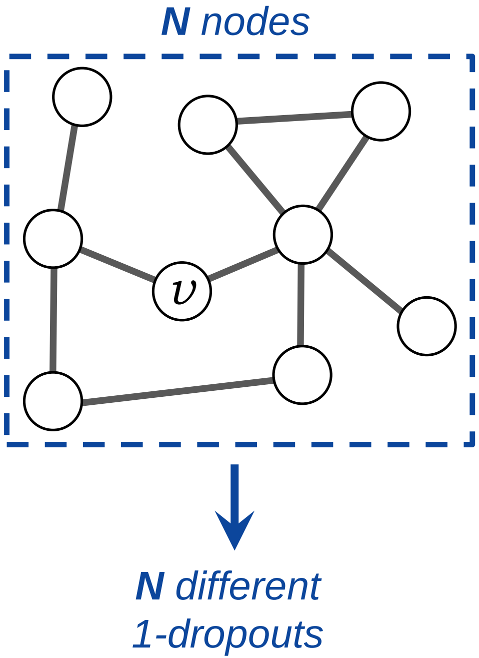
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



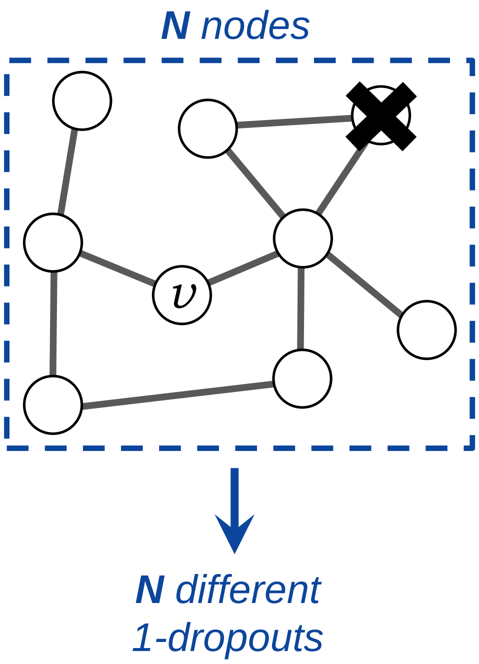
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



DropGNN with 1-dropouts

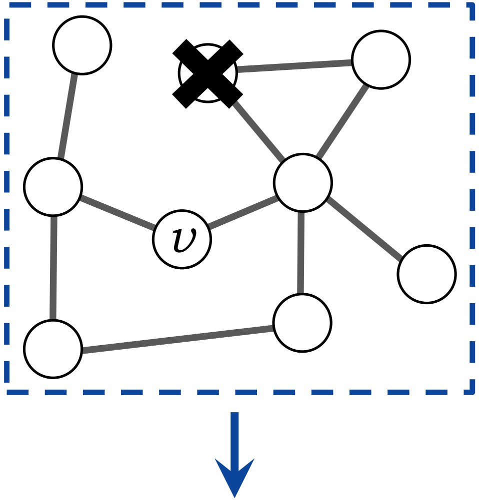
More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*

N *nodes*



N *different*
1-dropouts

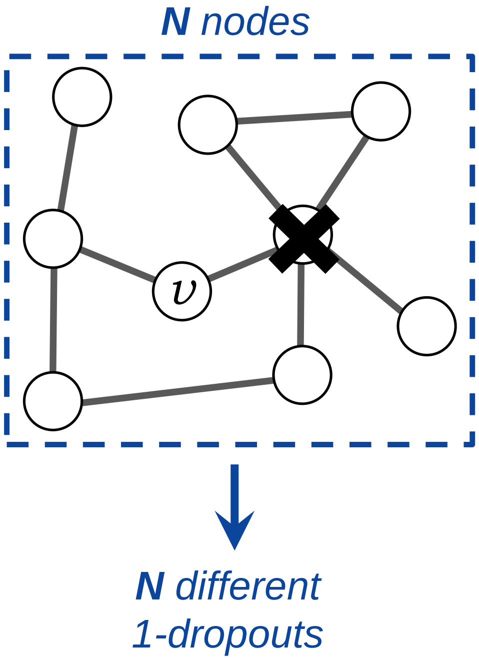
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



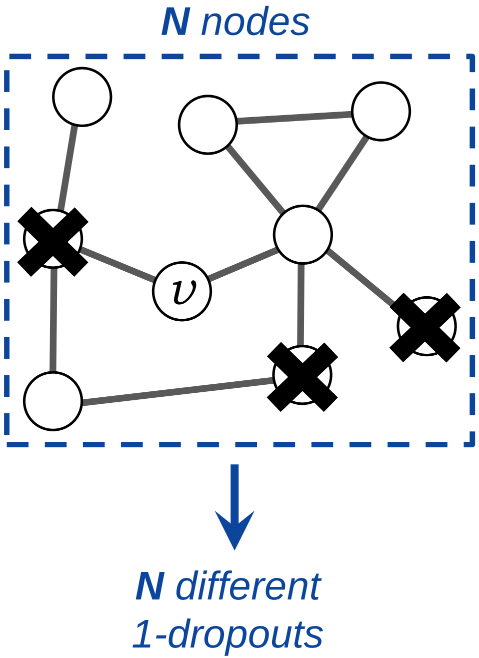
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



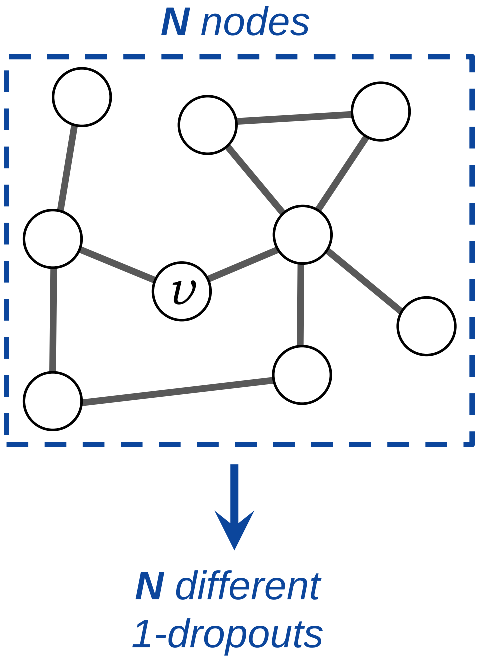
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



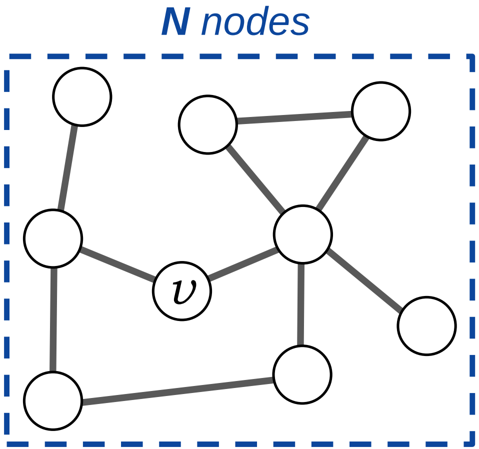
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

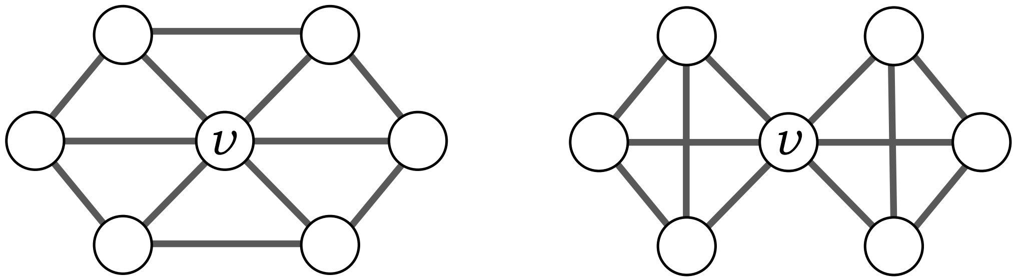
Observe every *1-dropout*



Theorem: if *#runs* ≈ $N \cdot \log N$, then we observe every 1-dropout with high probability.

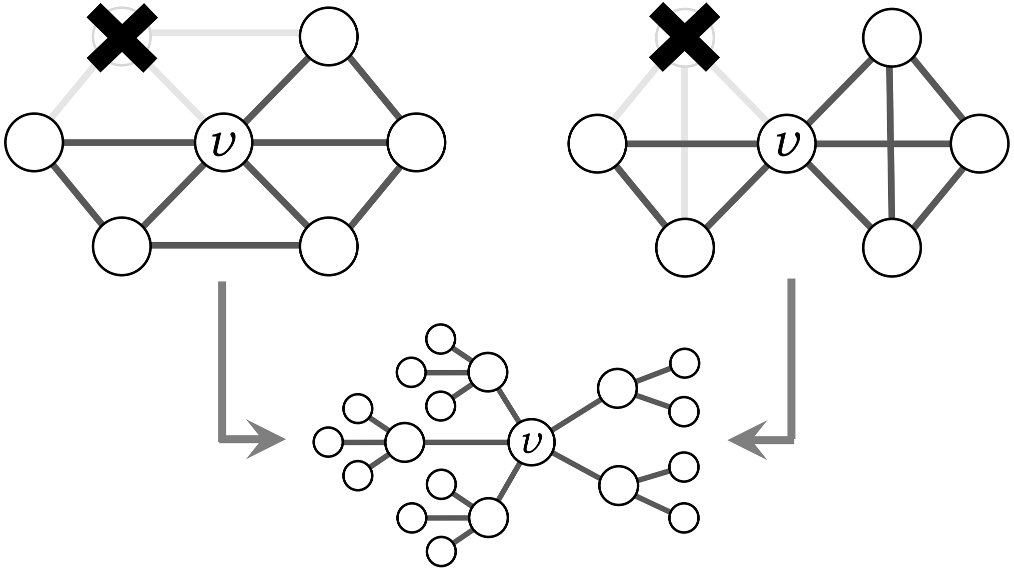
DropGNN with 1-dropouts

Theorem: There are graphs that cannot be distinguished from 1-dropouts only.



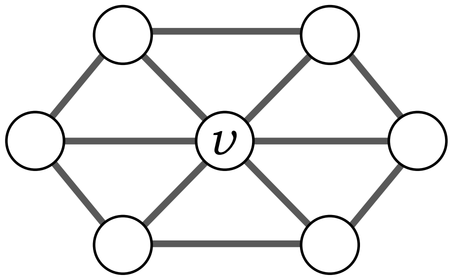
DropGNN with 1-dropouts

Theorem: There are graphs that cannot be distinguished from 1-dropouts only.



DropGNN with 1-dropouts

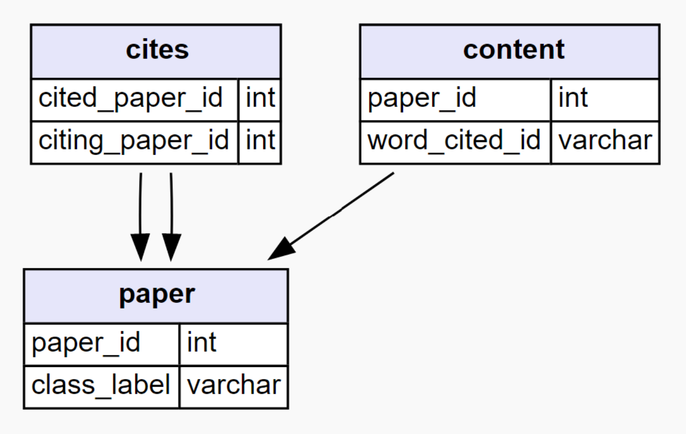
Theorem: There are graphs that cannot be distinguished from 1-dropouts only.

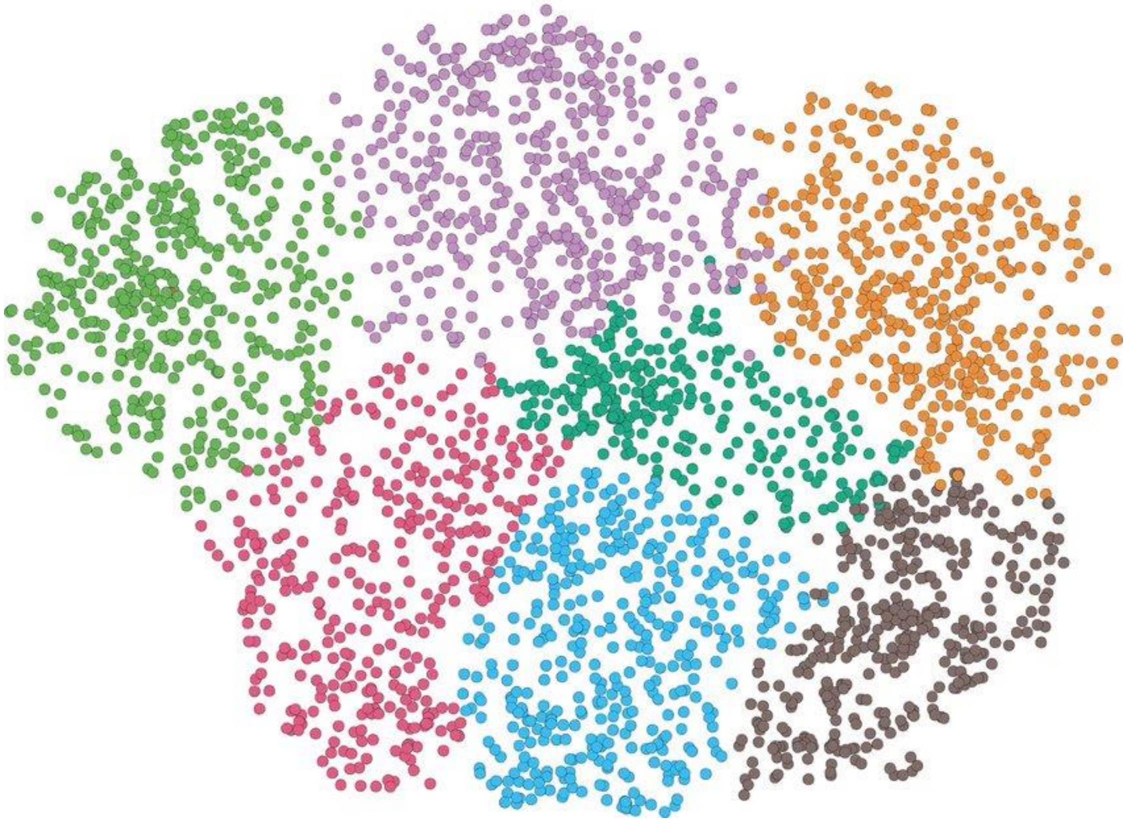


Theorem: in DropGNNs with *port numbers,* any two graphs can be distinguished from 1-dropouts.

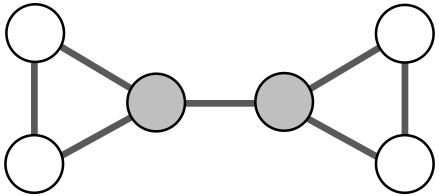

BAD DATA

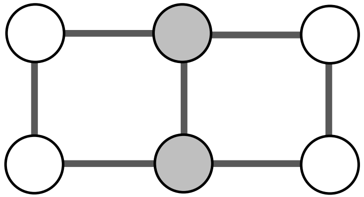
Example: CORA Benchmark





Example: CORA Benchmark





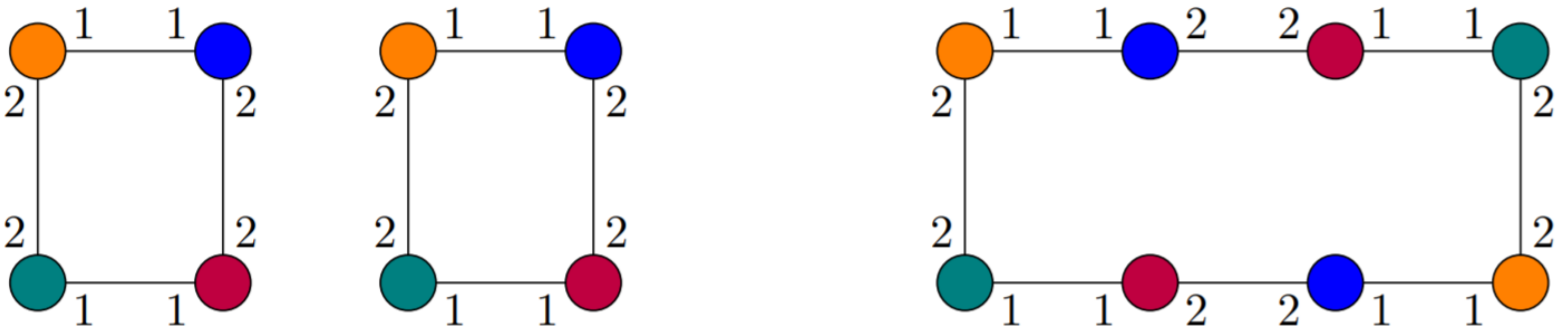
Title	Keywords	Neighbor Labels	Neighbor Keywords
Primes is in P	…	Crypto, …	…

Experiments: QM9 dataset

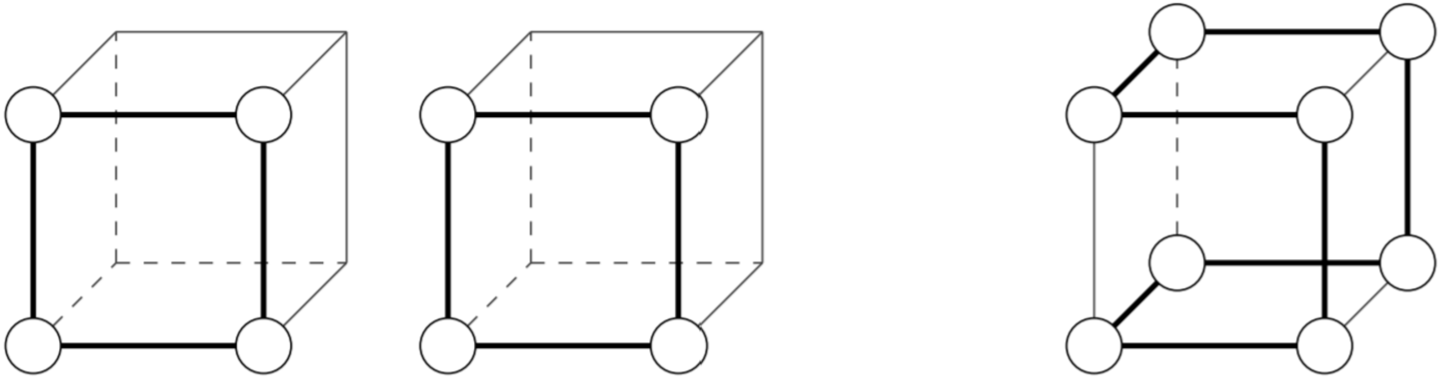
Property	Unit	GNN	DropGNN	PPGNN
μ	Debye	0.358	**0.077**	0.0934
α	$Bohr^3$	0.89	**0.238**	0.318
ϵ_{HOMO}	Hartree	0.00541	0.00235	**0.00174**
ϵ_{LUMO}	Hartree	0.00623	0.00241	**0.0021**
$\boldsymbol{\Delta\epsilon}$	Hartree	0.0066	0.0044	**0.0029**
$\langle R^2 \rangle$	$Bohr^2$	28.5	**0.472**	3.78
ZPVE	Hartree	0.00216	**0.000153**	0.000399
U_0	Hartree	2.05	0.251	**0.022**
U	Hartree	2.0	0.146	**0.0504**
H	Hartree	2.02	0.0845	**0.0294**
G	Hartree	2.02	**0.188**	0.24
C_v	cal/(mol K)	0.42	0.0740	**0.0144**

Other Extension Ideas?

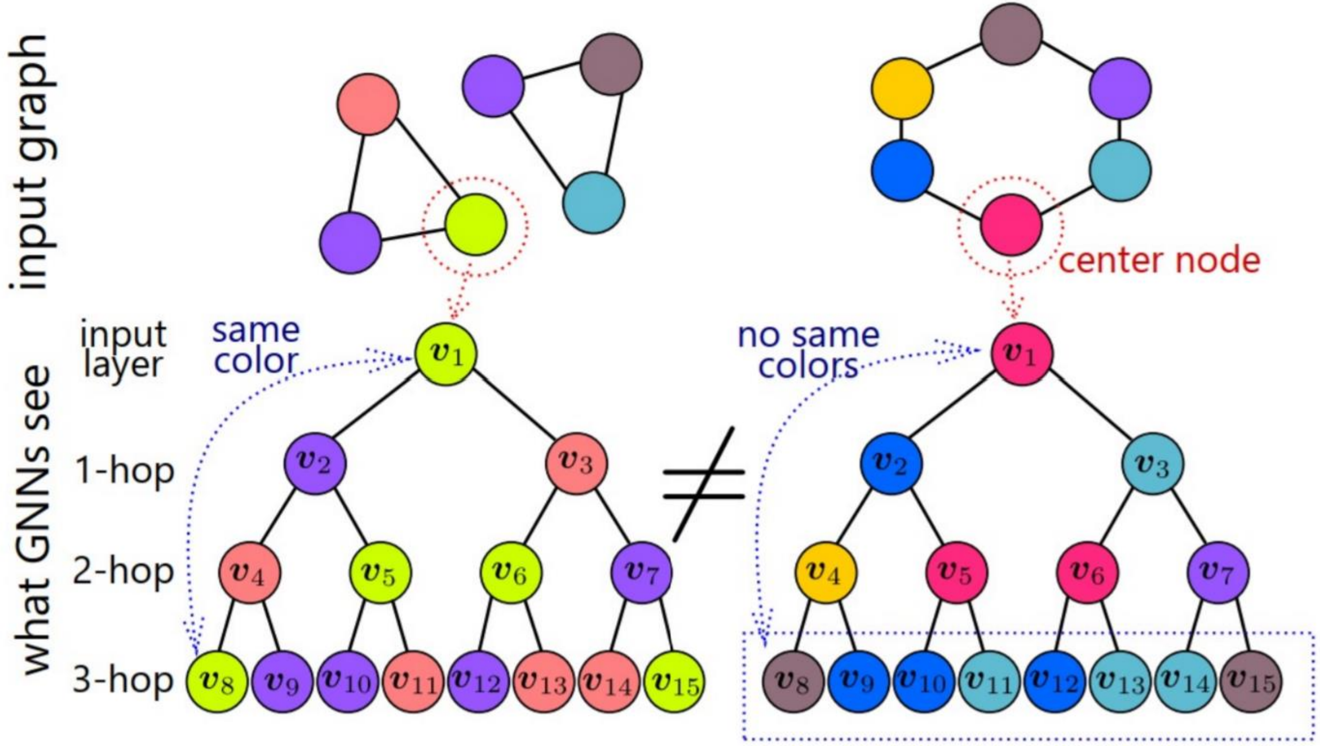
Port Numbers



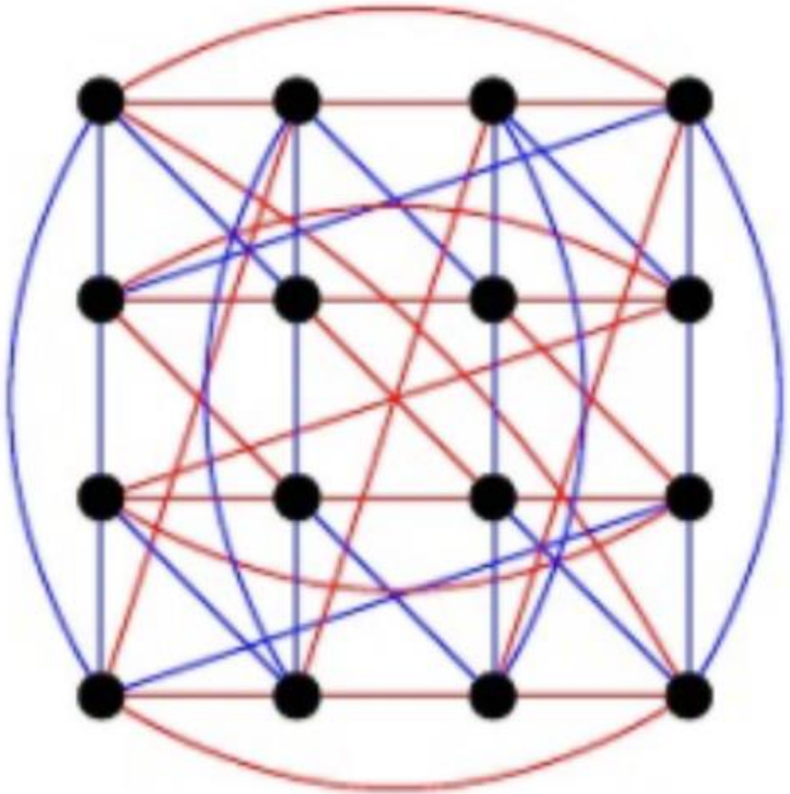
Angle Features

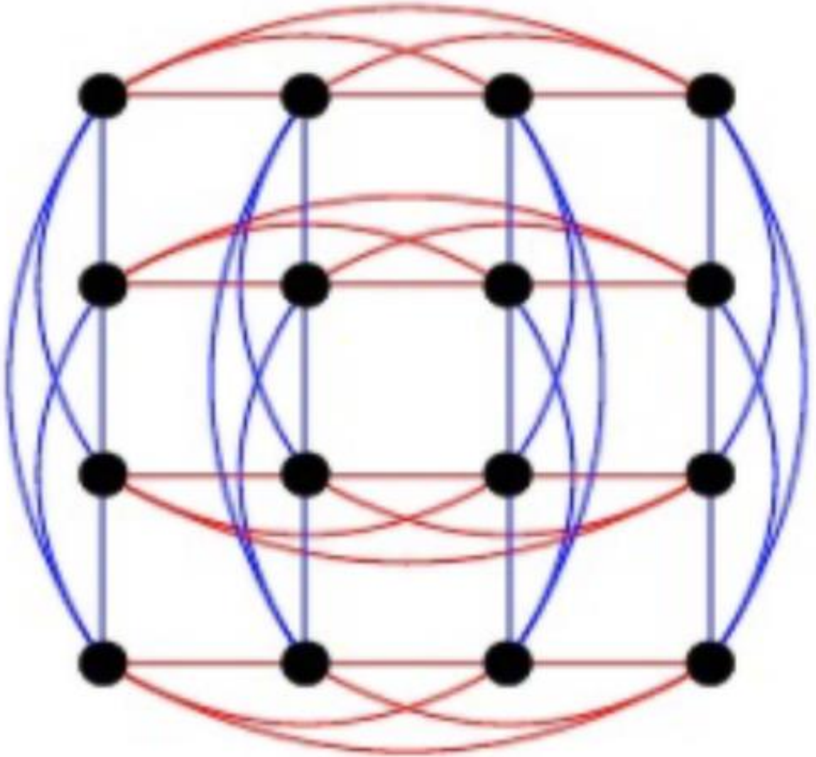


Random Features

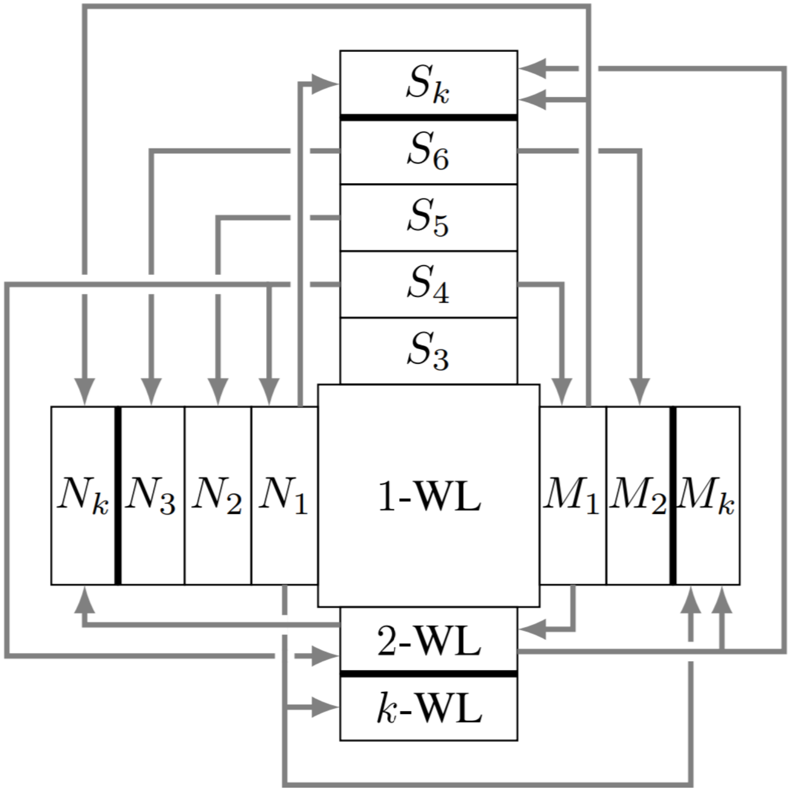


2-WL





Comparisons of Extensions



Open Questions

- **Theory:** characterization of graphs that can be distinguished by extensions?
- **Experiments:** other applications where the graph structure is crucial?
- **General:** similar approach in other deep learning areas?

Thank You!

Questions & Comments?