From Distributed Algorithms to Machine Learning and Back

ML + GRAPHS = PODC?

Roger Wattenhofer


Midjourney


Boris Eldagsen

write a poem about traffic lights

In the heart of a bustling city's night,
A sentinel stands, glowing with light,
With colors of red, yellow, and green,
A triad of hues, a silent machine.

A beacon of order midst chaos and speed,
Guiding the masses, an unspoken creed,

From Distributed Algorithms to Machine Learning and Back

Roger Wattenhofer

Deep Learning is Robust to Massive Label Noise

David Rolnick [*1] Andreas Veit [*2] Serge Belongie [2] Nir Shavit [3]

Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent

Peva Blanchard
EPFL, Switzerland
peva.blanchard@epfl.ch

El Mahdi El Mhamdi[*]
EPFL, Switzerland
elmahdi.elmhamdi@epfl.ch

Rachid Guerraoui
EPFL, Switzerland
rachid.guerraoui@epfl.ch

Julien Stainer
EPFL, Switzerland
julien.stainer@epfl.ch

QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding

Dan Alistarh
IST Austria & ETH Zurich
dan.alistarh@ist.ac.at

Demjan Grubic
ETH Zurich & Google
demjangrubic@gmail.com

Jerry Z. Li
MIT
jerryzli@mit.edu

Ryota Tomioka
Microsoft Research
ryoto@microsoft.com

Milan Vojnovic
London School of Economics
M.Vojnovic@lse.ac.uk

Byzantine Fault-Tolerant Distributed Machine Learning using D-SGD and Norm-Based Comparative Gradient Elimination (CGE)

Nirupam Gupta
EPFL
Lausanne, Switzerland
nirupam115@gmail.com

Shuo Liu
Georgetown University
Washington, D.C., USA
sl1539@georgetown.edu

Nitin Vaidya
Georgetown University
Washington, D.C., USA
nv198@georgetown.edu

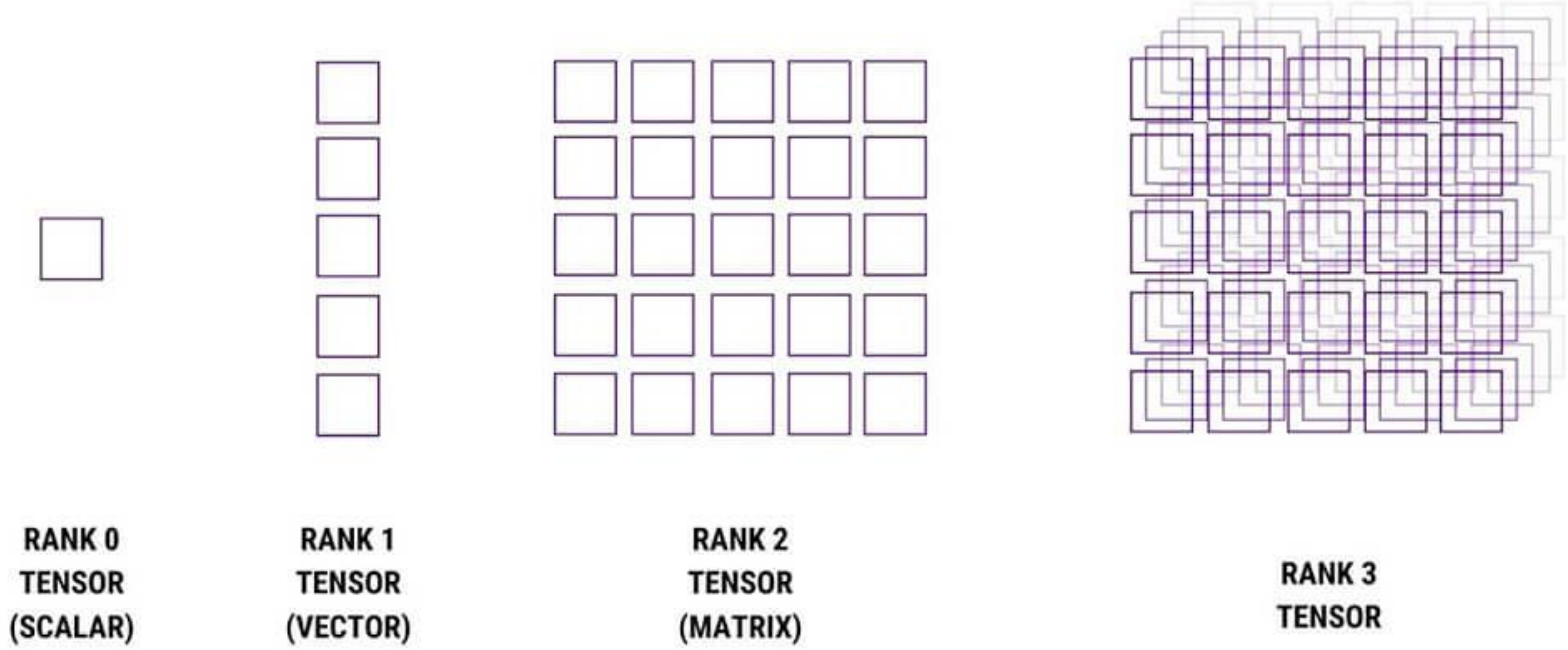
Concurrency
& Consensus

Byzantine
Federated

Graph
Algorithms

This
Talk

Machine Learning Deals with …



Networks

Social Networks

Neural Networks

Mobile Networks

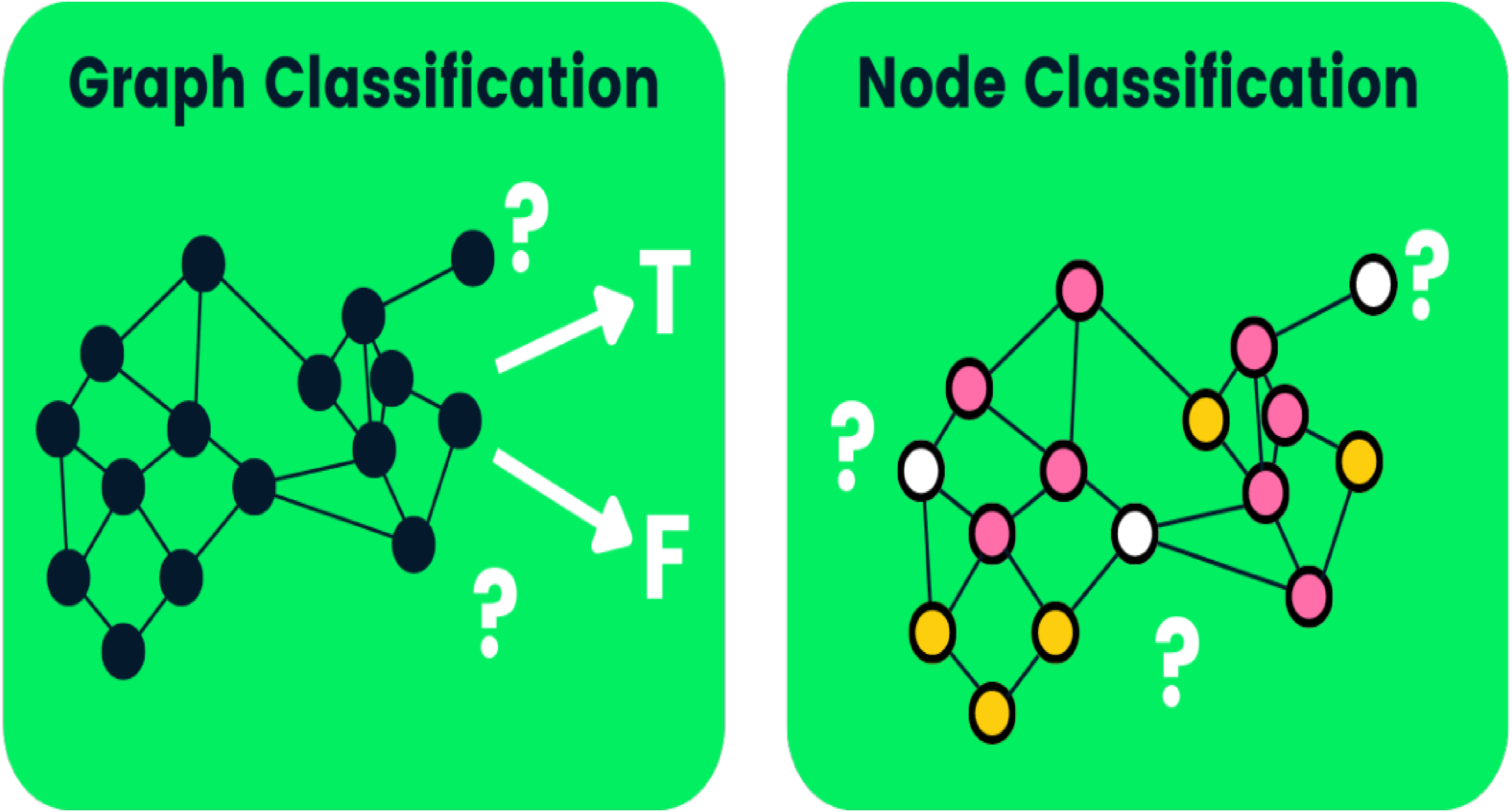
Wireless Networks

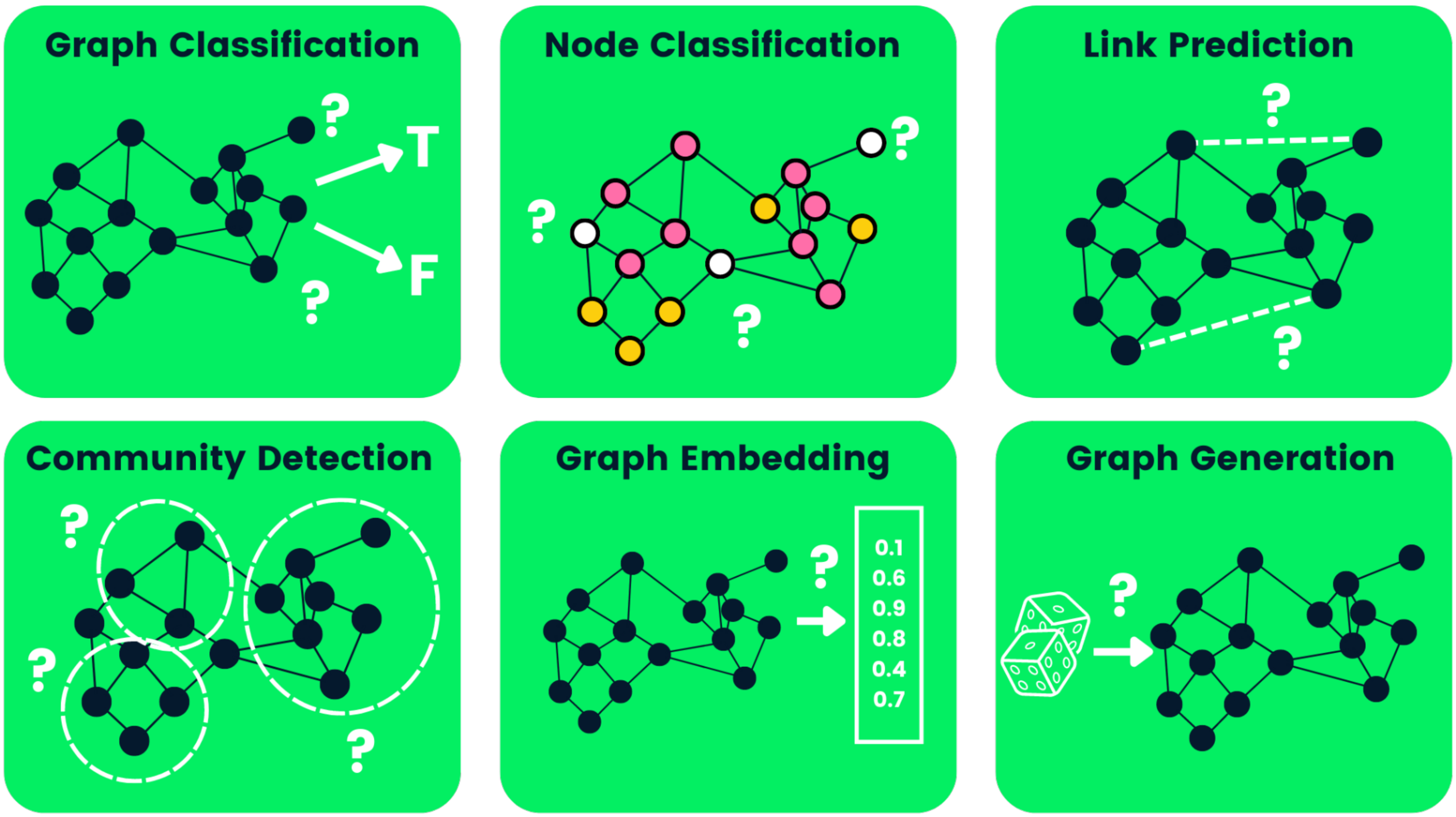
Financial Networks

Economic Networks

Biological Networks

Computer Networks


Graph Classification
?
T
F
?
Node Classification
?
?
?


Graph Classification
?
T
F
?
Node Classification
?
?
?
Link Prediction
?
?
Community Detection
?
?
?
Graph Embedding
?
0.1
0.6
0.9
0.8
0.4
0.7
Graph Generation
?

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

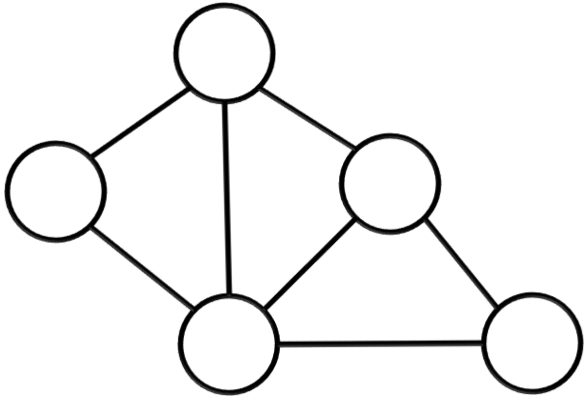
Abstract. The paper provides an introduction into the theoretical expressiveness of graph neural networks. We discuss the basic properties and main applications of standard GNN models, and we show how these constructions are both upper and lower bounded in expressive power by the Weisfeiler-Lehman test. 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.

GNNs vs. Distributed Computing

Distributed Computing (Message Passing)

Nodes communicate with neighbors by sending messages.

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

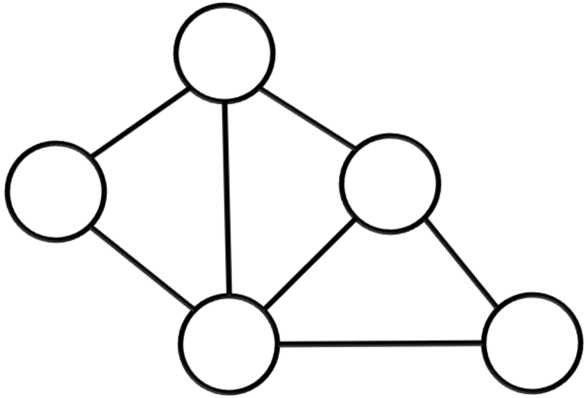


```
each round:
every node:
1. send msgs
2. rcv msgs
3. compute
```

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

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

ML Track

"Learned" parameters

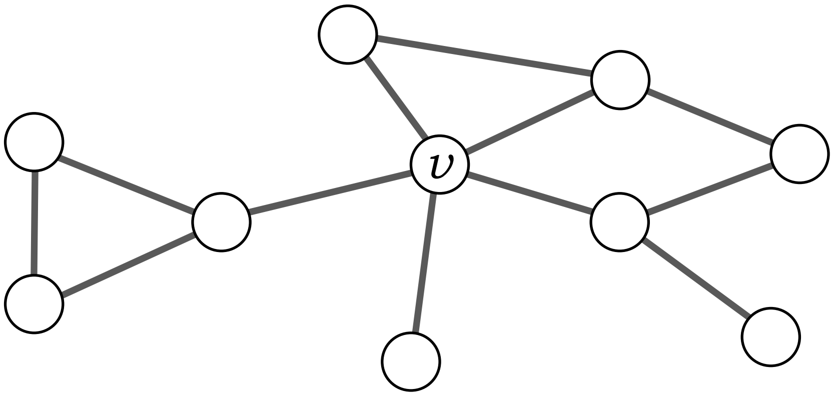
Usually, node features

Aggregated messages

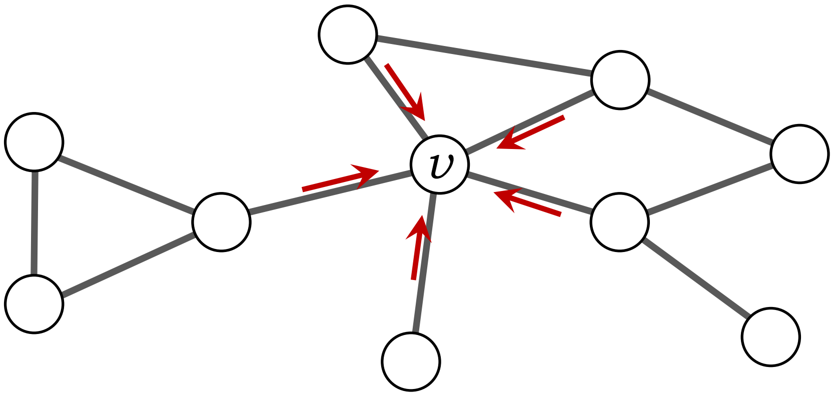
Solve classification (node, edge, graph)

More Details, Please!

Graph Neural Networks

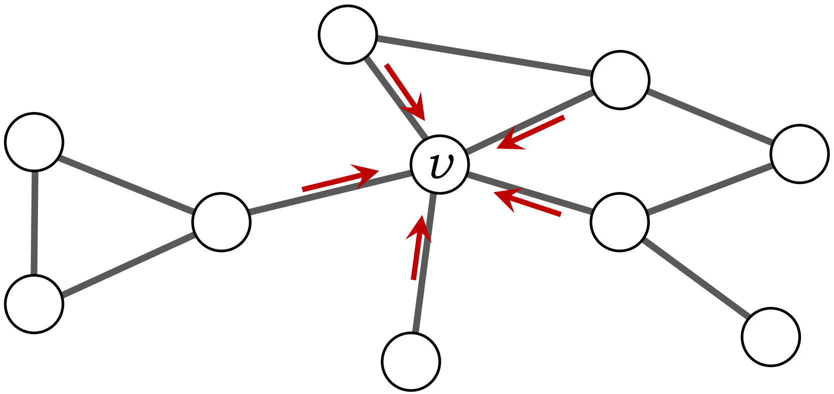


Graph Neural Networks



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

Graph Neural Networks

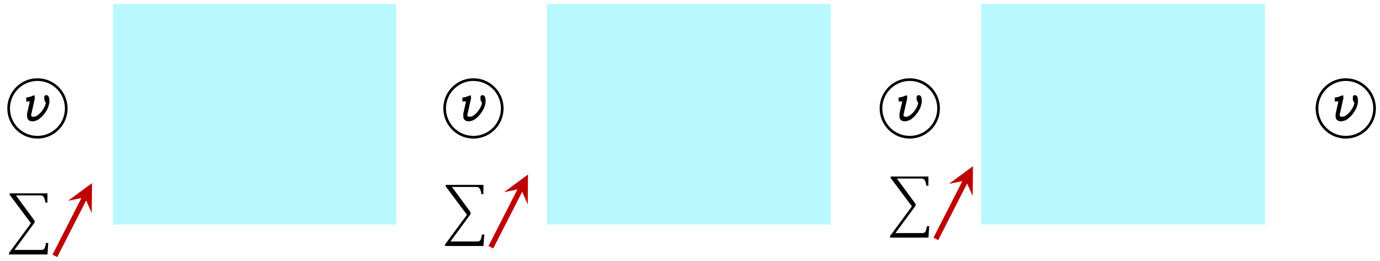


$a_v = \text{AGGREGATE}\left(\{\{\, h_u \mid u \in N(v) \,\}\}\right)$ (Min, Max, Mean, Sum)

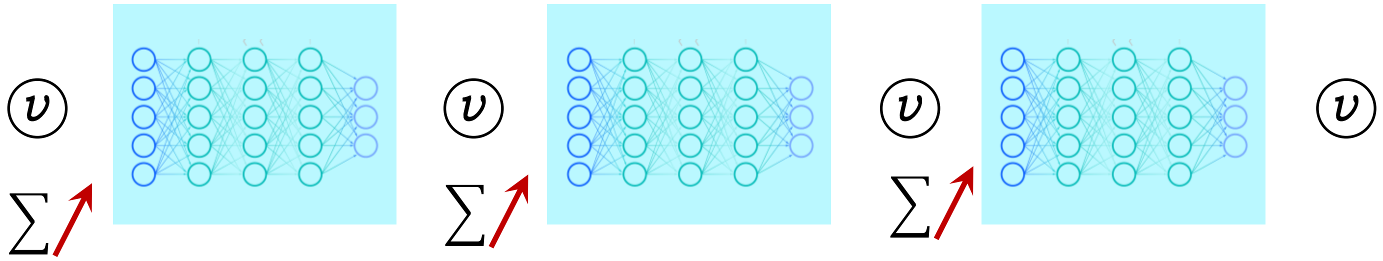
$h_v^{(t+1)} = \text{UPDATE}\left(h_v, a_v\right)$

Graph Neural Networks

Graph Neural Networks

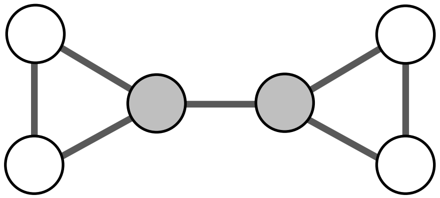


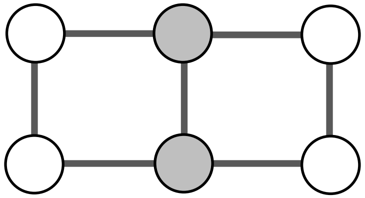
Graph Neural Networks



GNN Limitations?

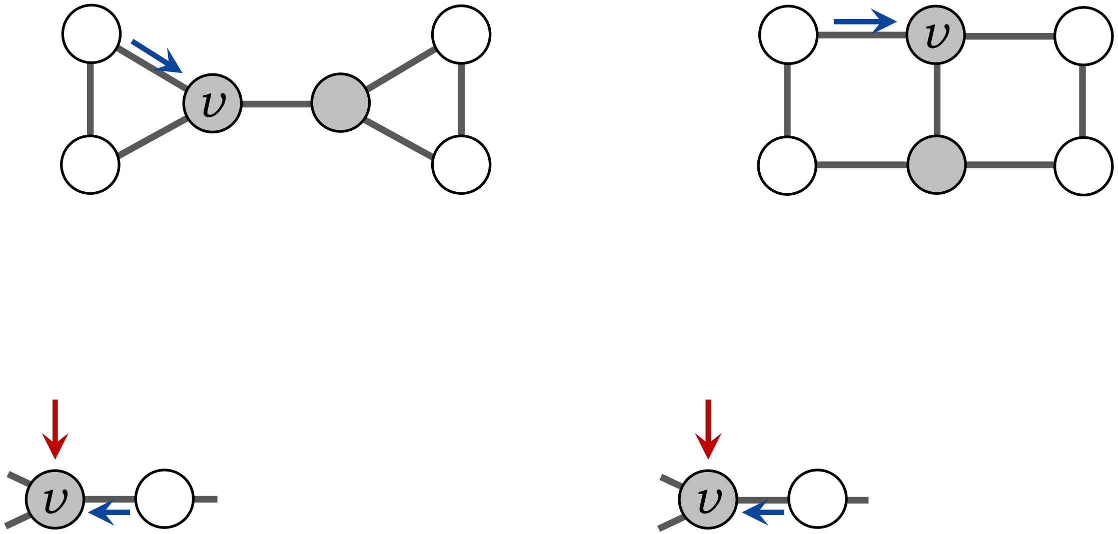
Limits of GNNs



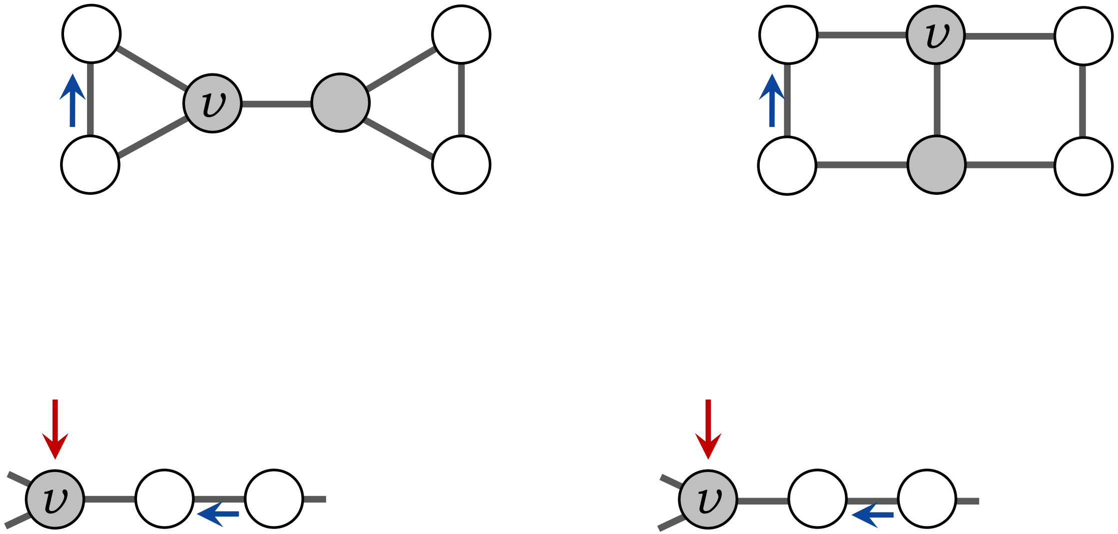


Limits of GNNs

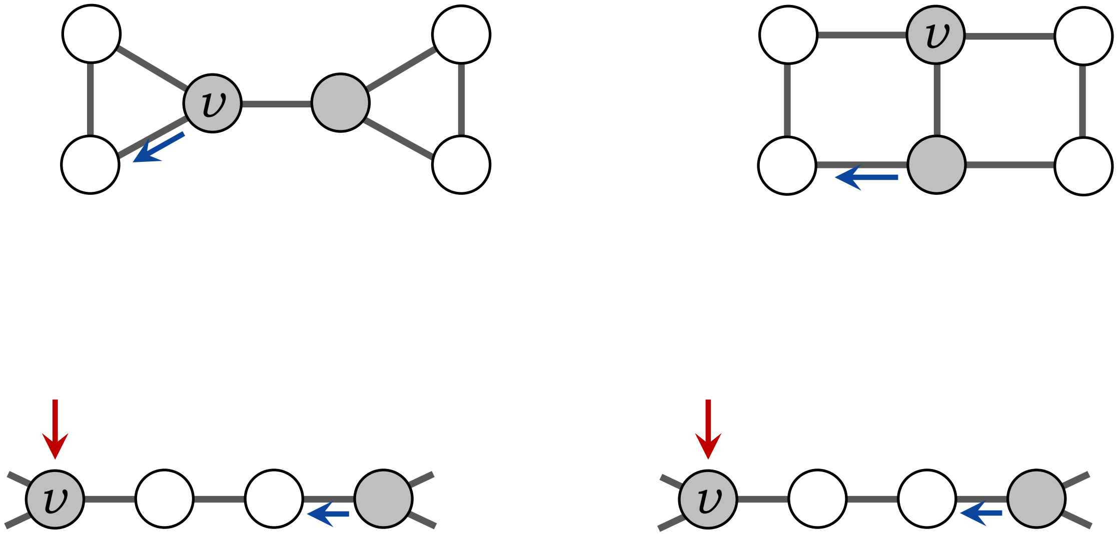
Limits of GNNs



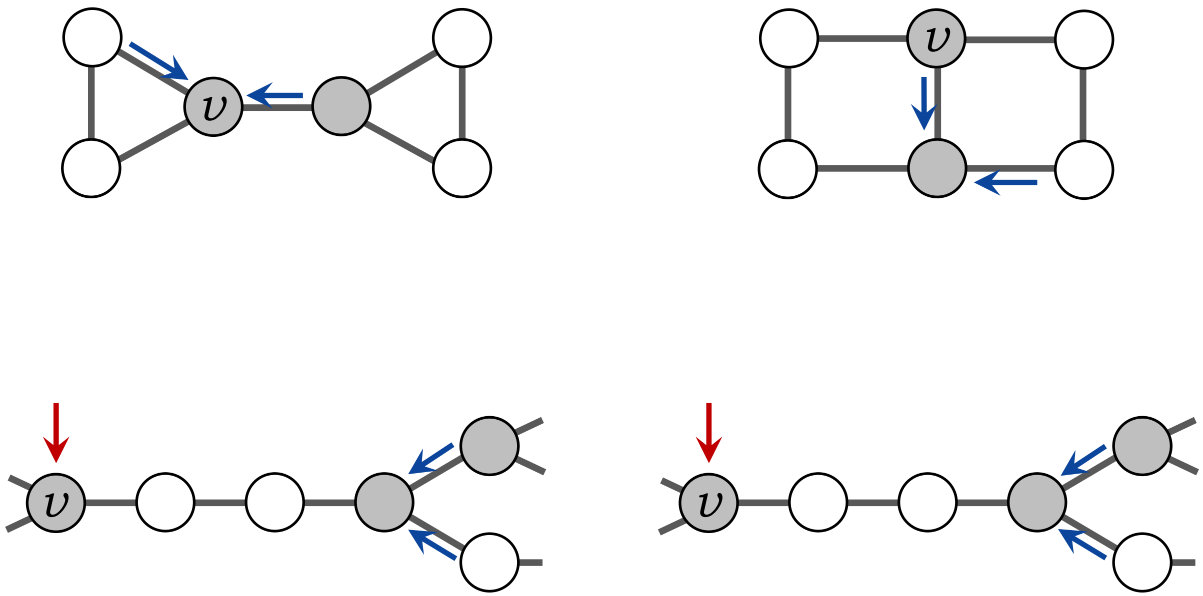
Limits of GNNs



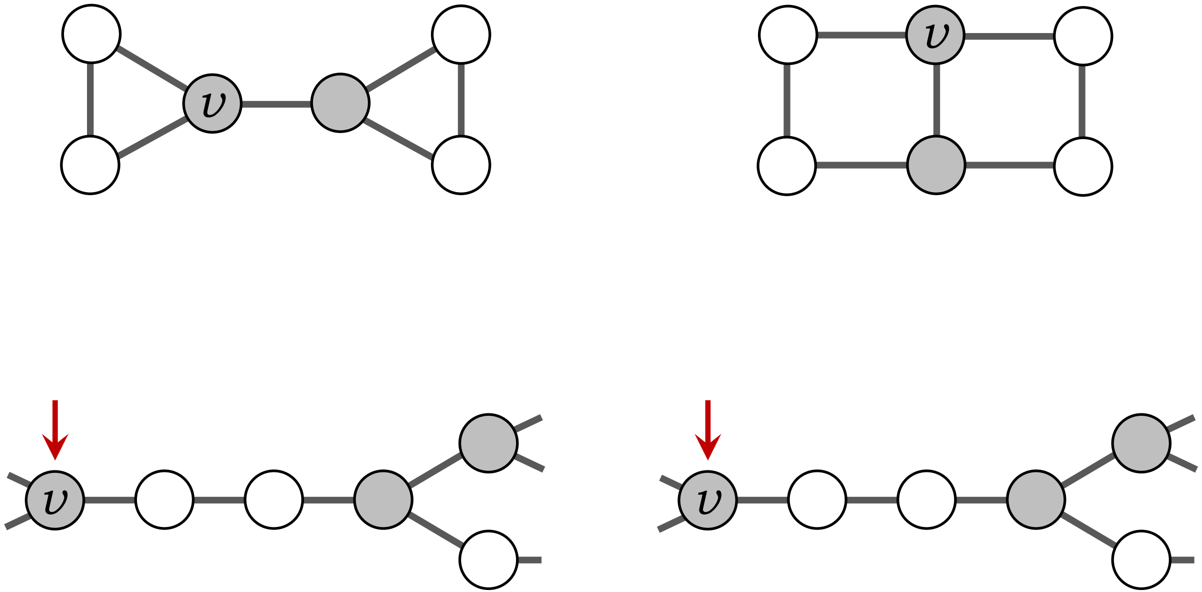
Limits of GNNs



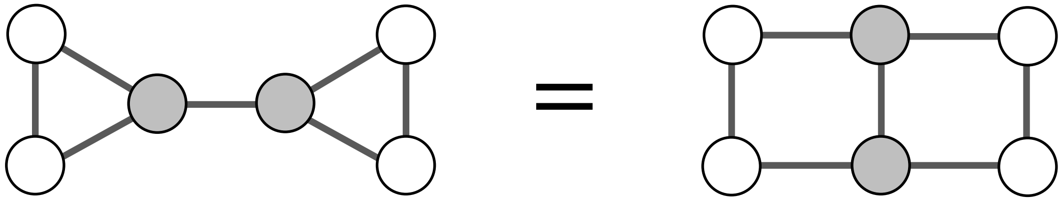
Limits of GNNs



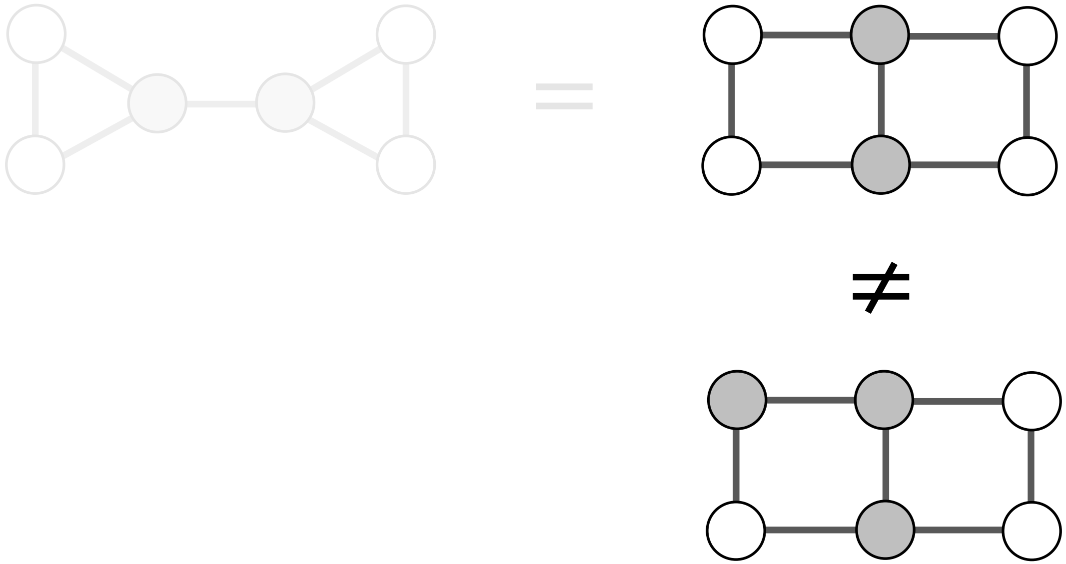
Limits of GNNs



Graph Neural Networks

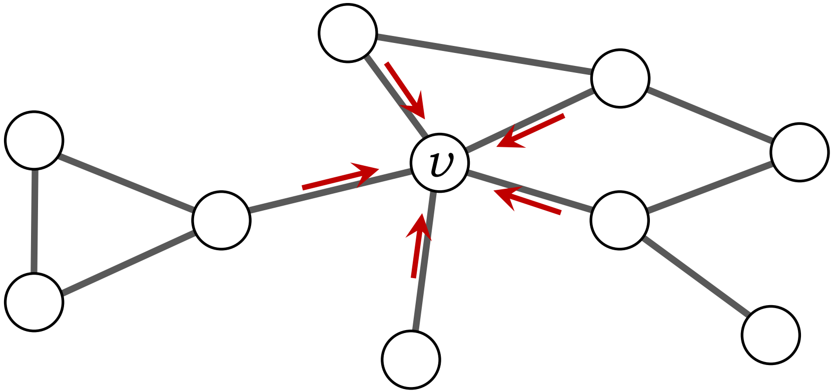


Graph Neural Networks



Graph Neural Networks

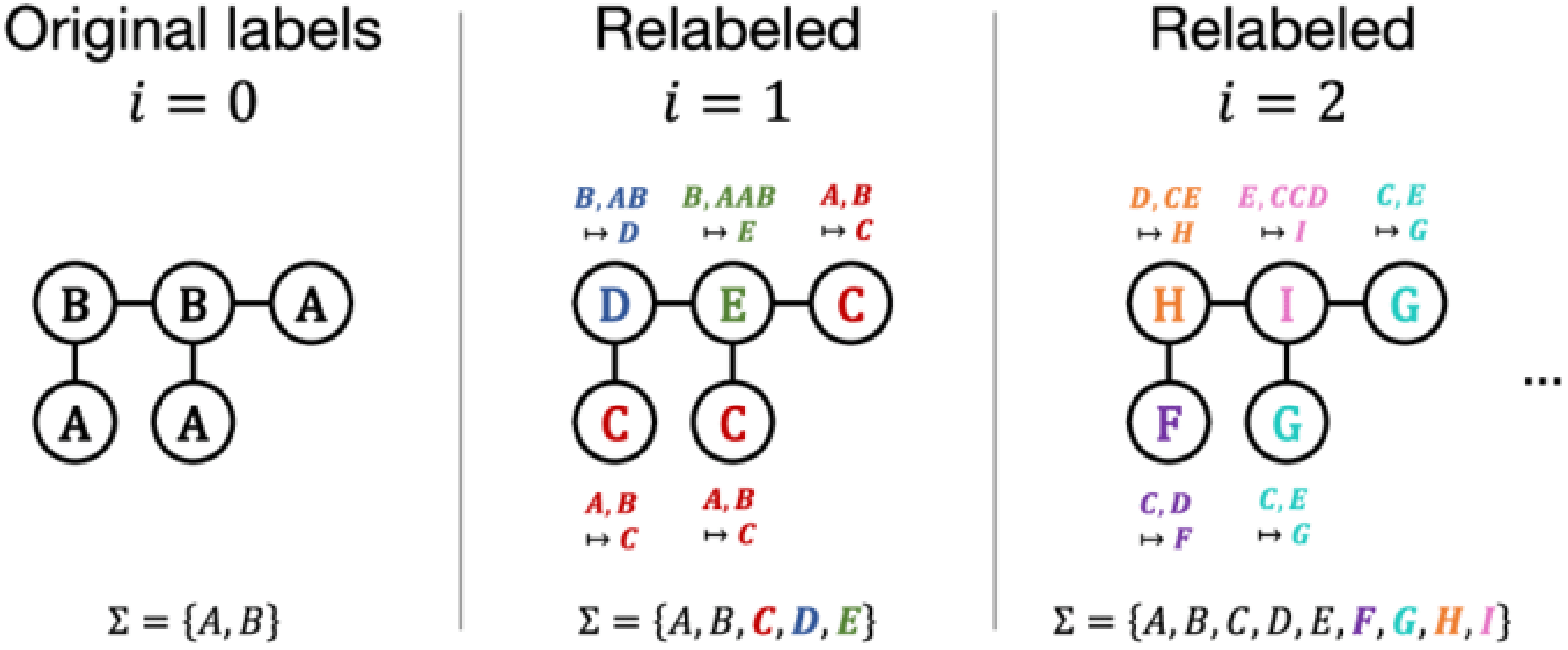
GNNs $\leq$ Weisfeiler-Lehman test



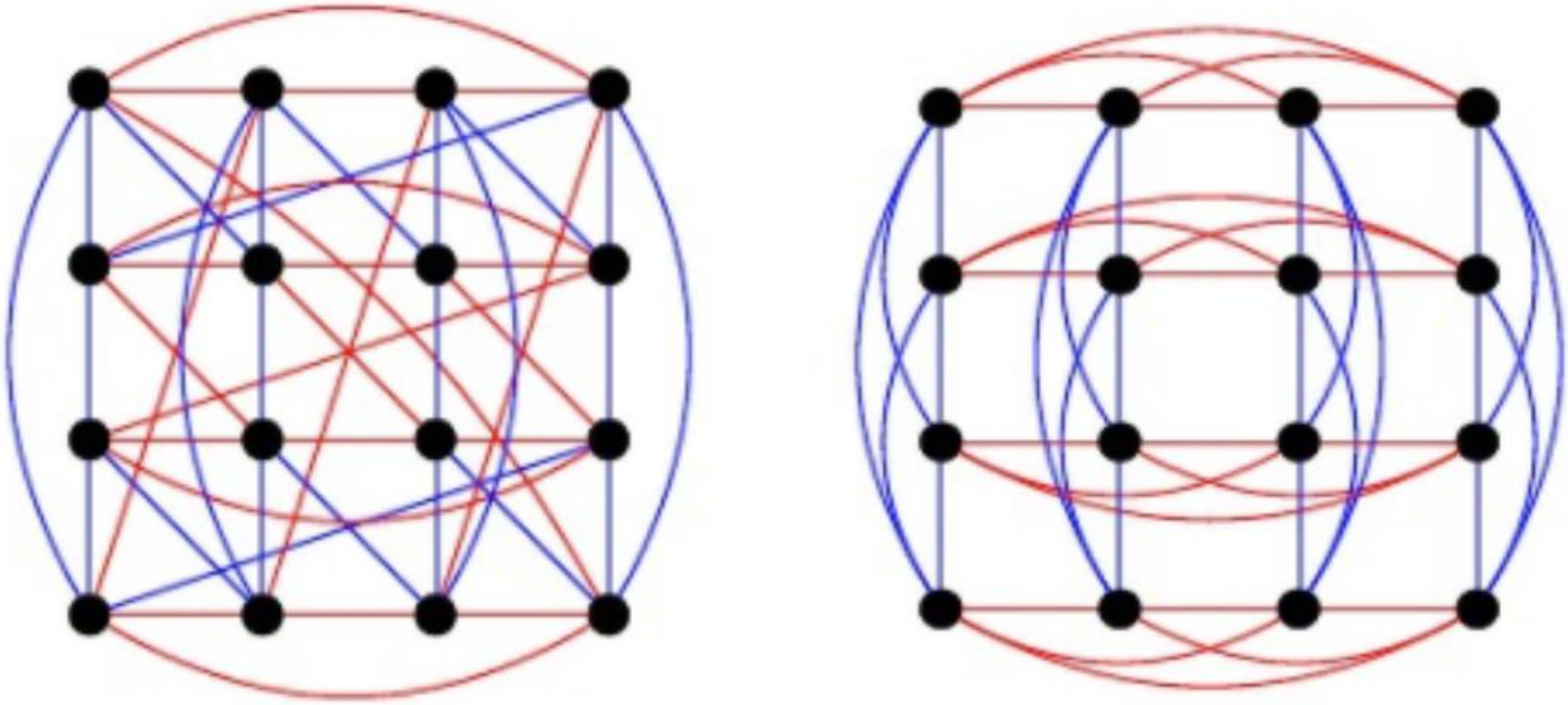
$$a_v = \text{AGGREGATE}\left(\{\{\, h_u \mid u \in N(v) \,\}\}\right)$$

$$h_v^{(t+1)} = \text{UPDATE}\left(h_v, a_v\right)$$

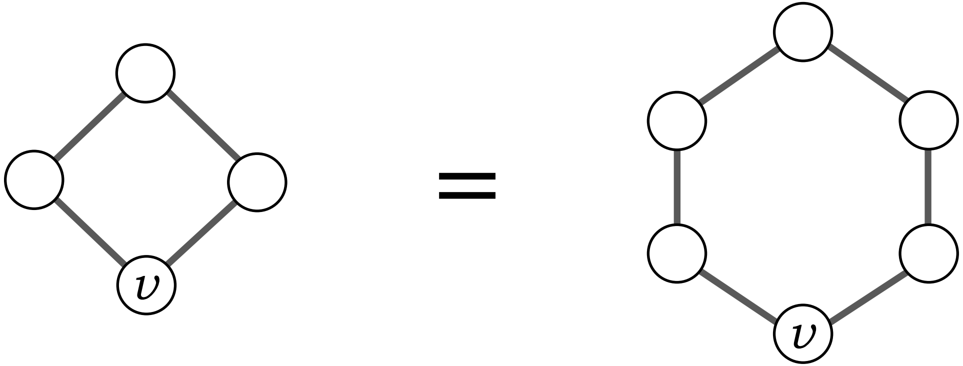
Weisfeiler-Lehman Graph Isomorphism Test



Shrikande vs. Rooks



GNNs Fail on e.g. Cycles



DC Track

anonymous

local

congest

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

ML Track

oversmoothing

underreaching

oversquashing

More Expressive GNNs?

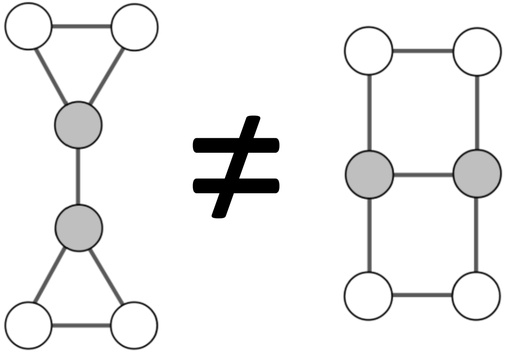
DropGNN: Random Dropouts Increase the Expressiveness of Graph Neural Networks

Pál András Papp
ETH Zurich
apapp@ethz.ch

Karolis Martinkus
ETH Zurich
martinkus@ethz.ch

Lukas Faber
ETH Zurich
lfaber@ethz.ch

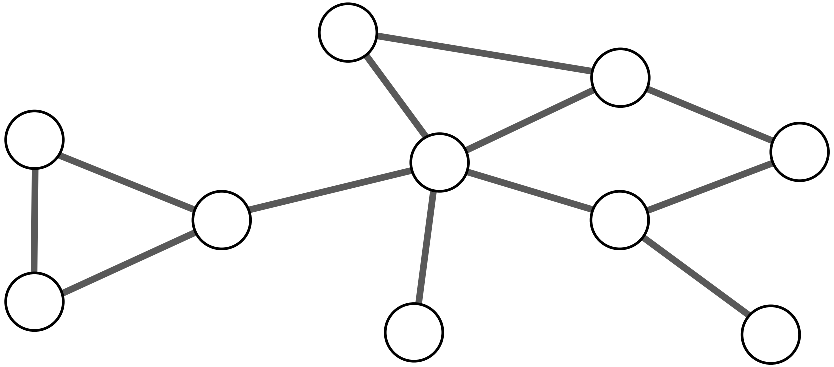
Roger Wattenhofer
ETH Zurich
wattenhofer@ethz.ch



GNNs with Dropouts

Multiple runs of the GNN

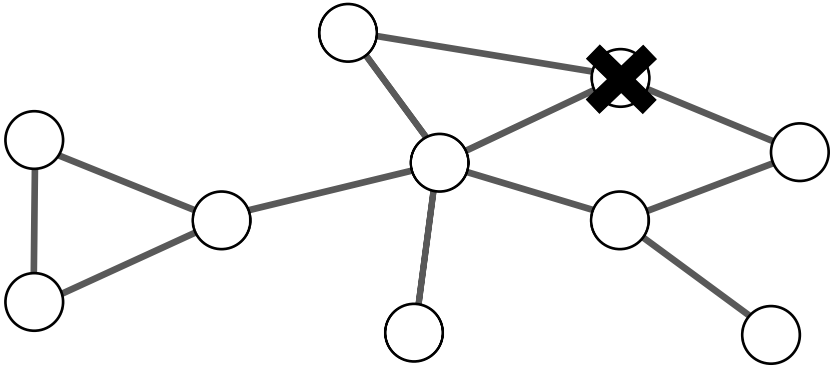
Each node removed with probability *p* independently



GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

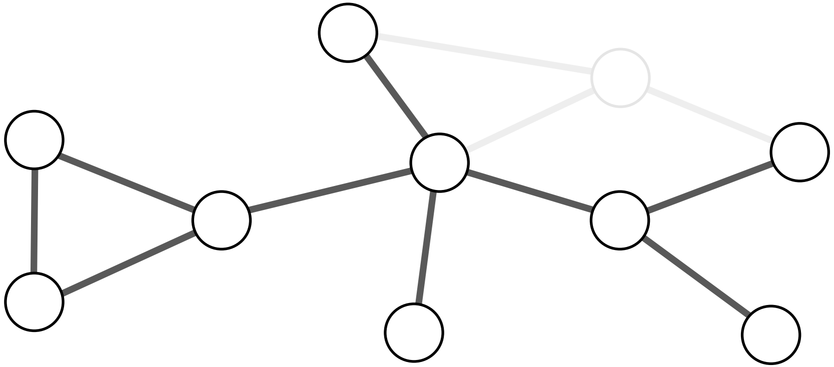


Run #1

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

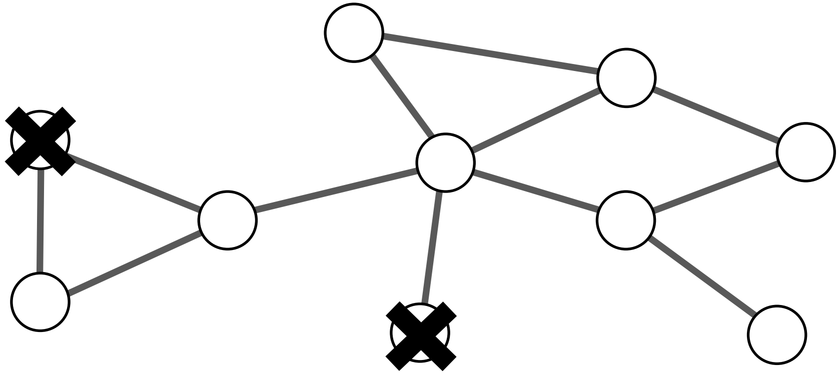


Run #1

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

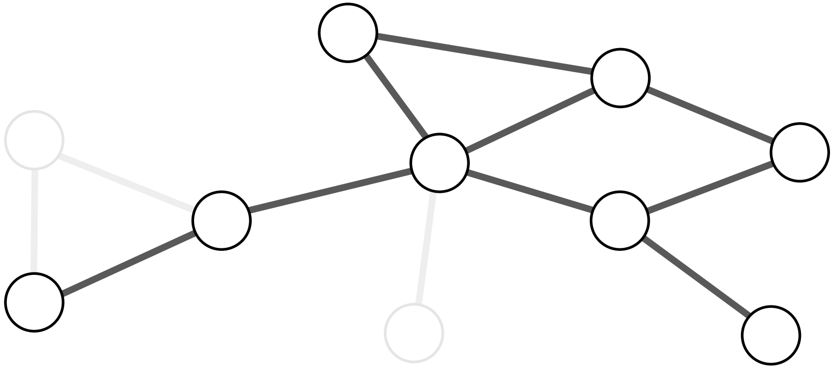


Run #2

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

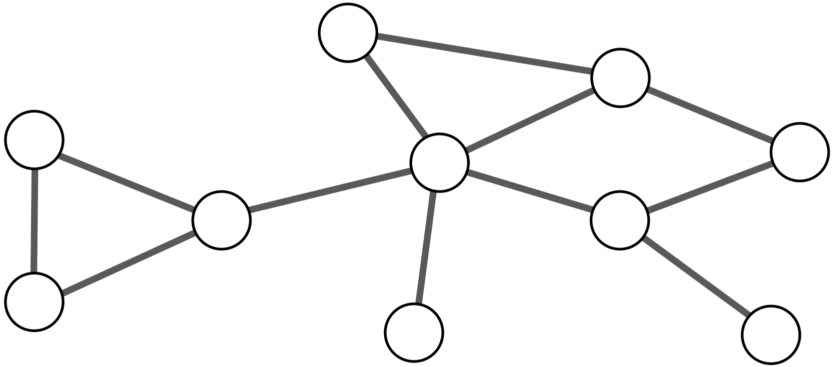


Run #2

GNNs with Dropouts

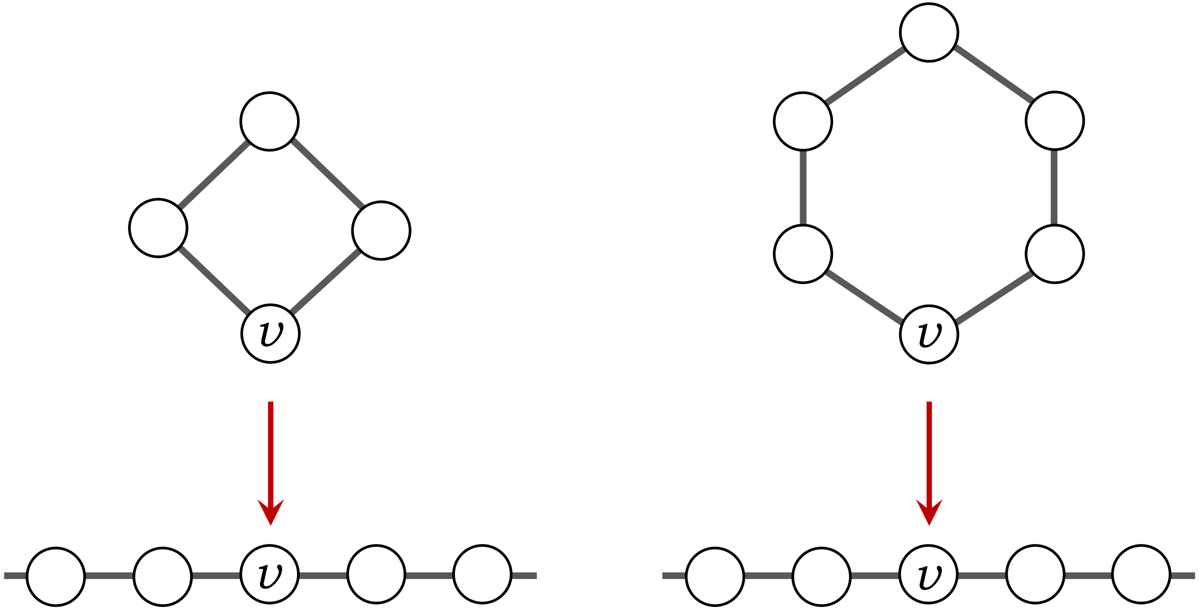
Multiple runs of the GNN

Each node removed with probability *p* independently

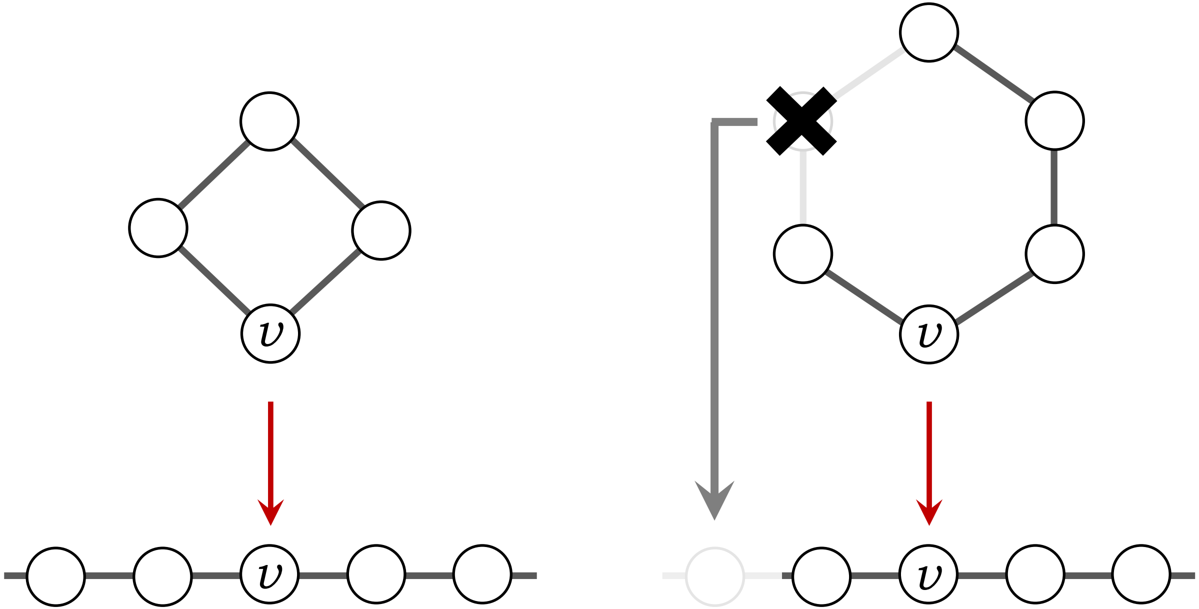


Run #3

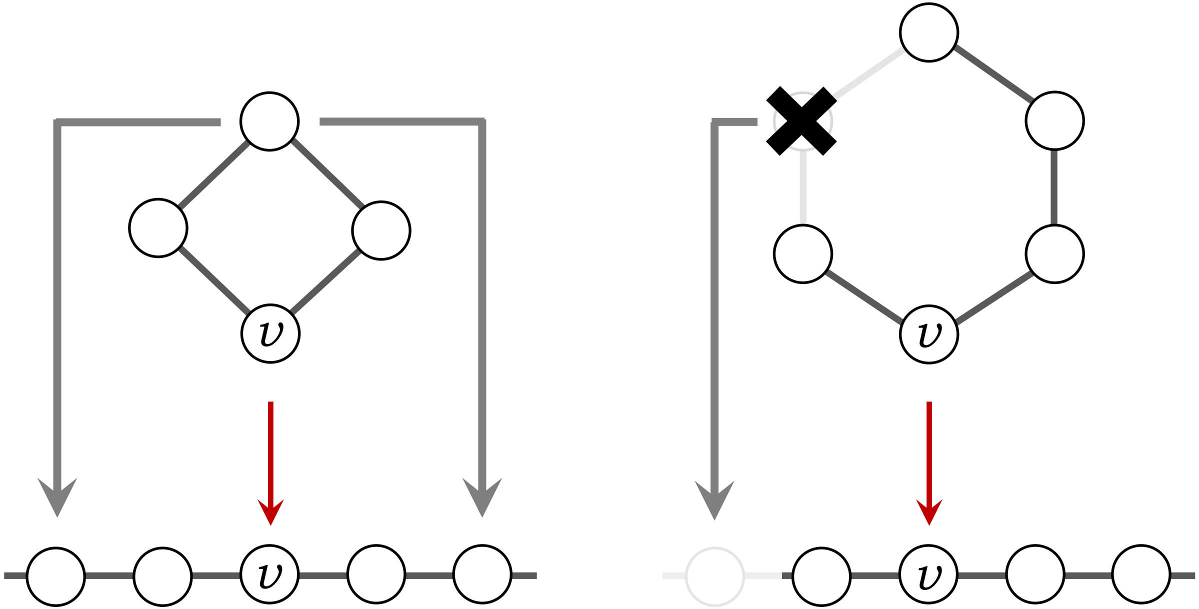
GNNs with Dropouts



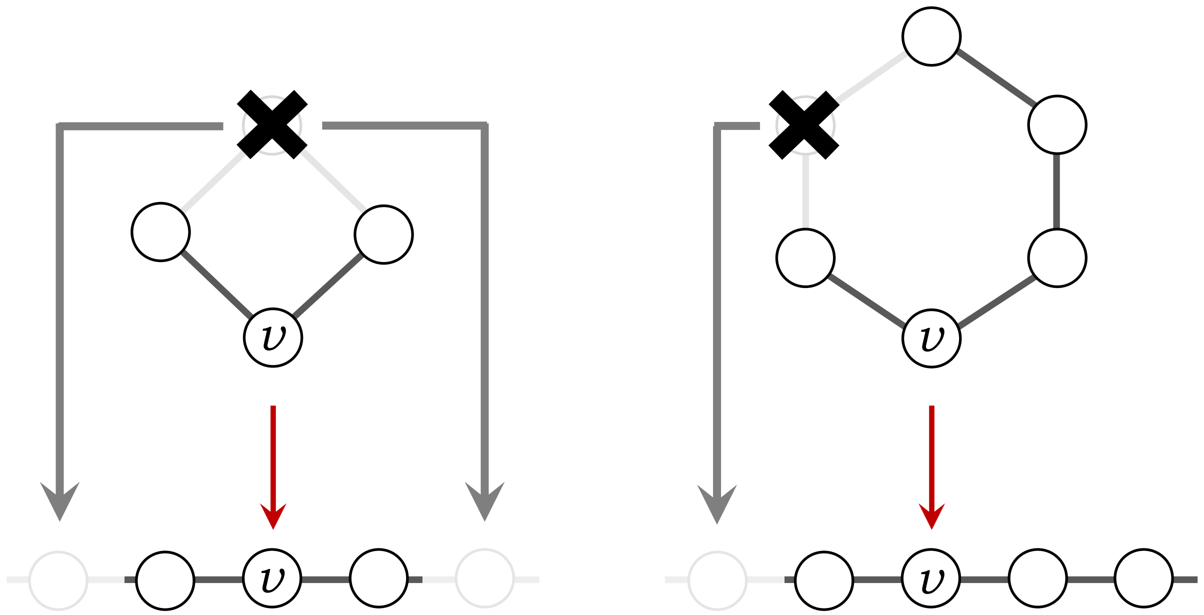
GNNs with Dropouts



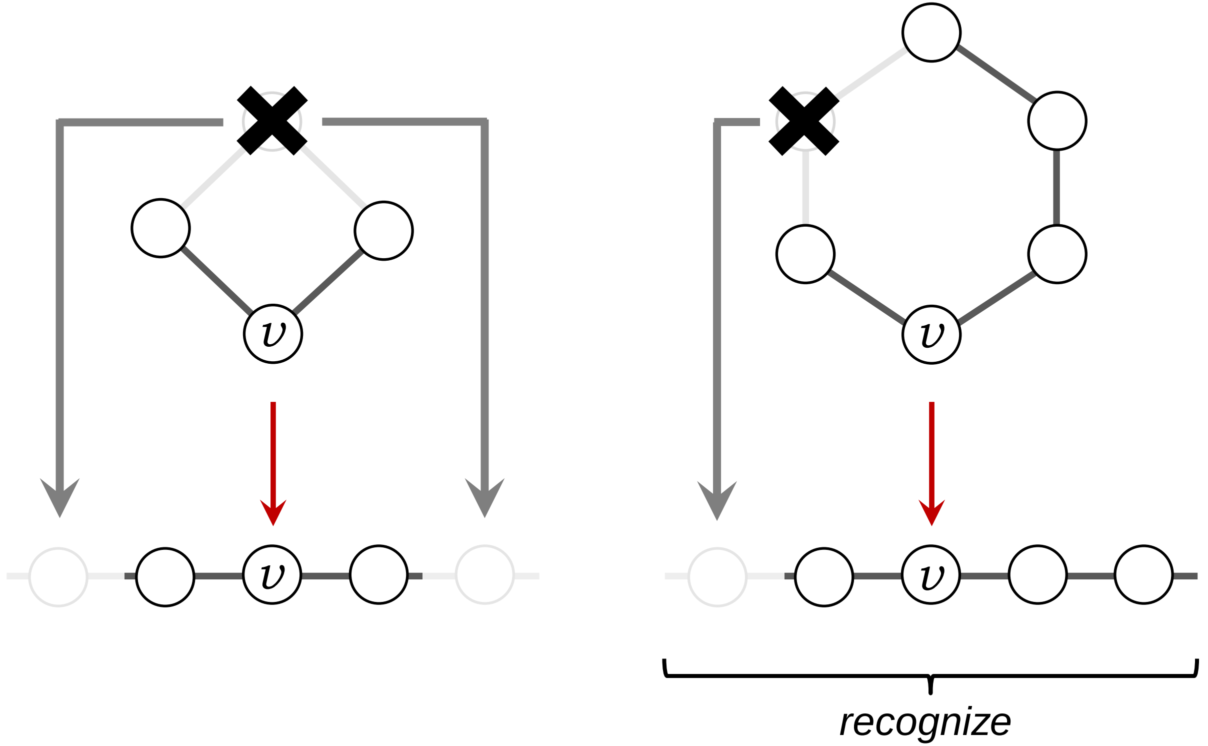
GNNs with Dropouts



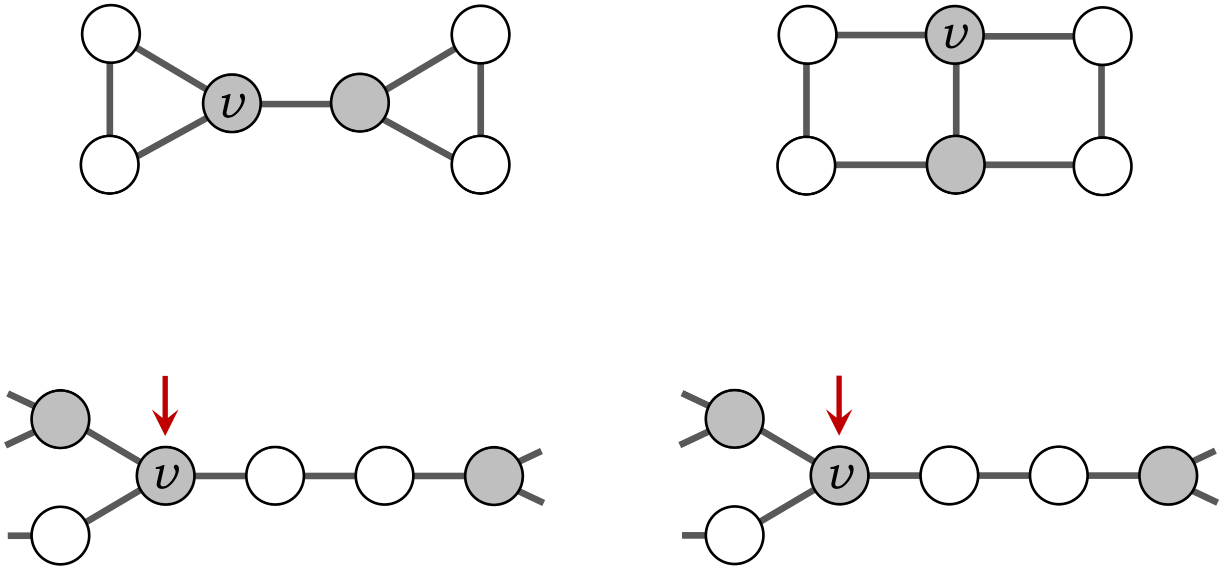
GNNs with Dropouts



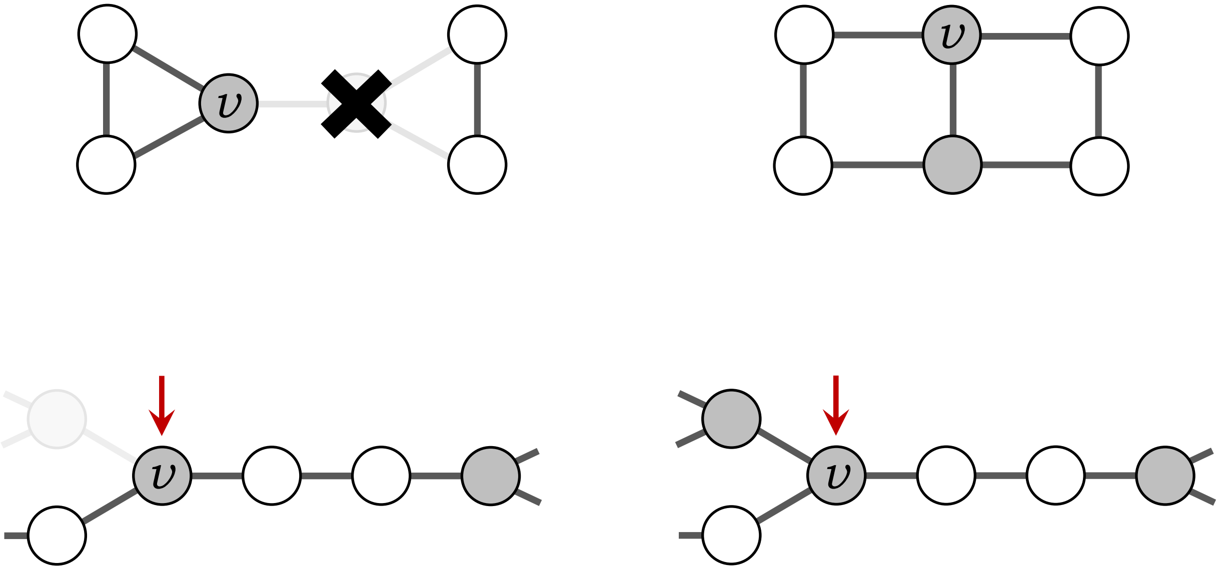
GNNs with Dropouts



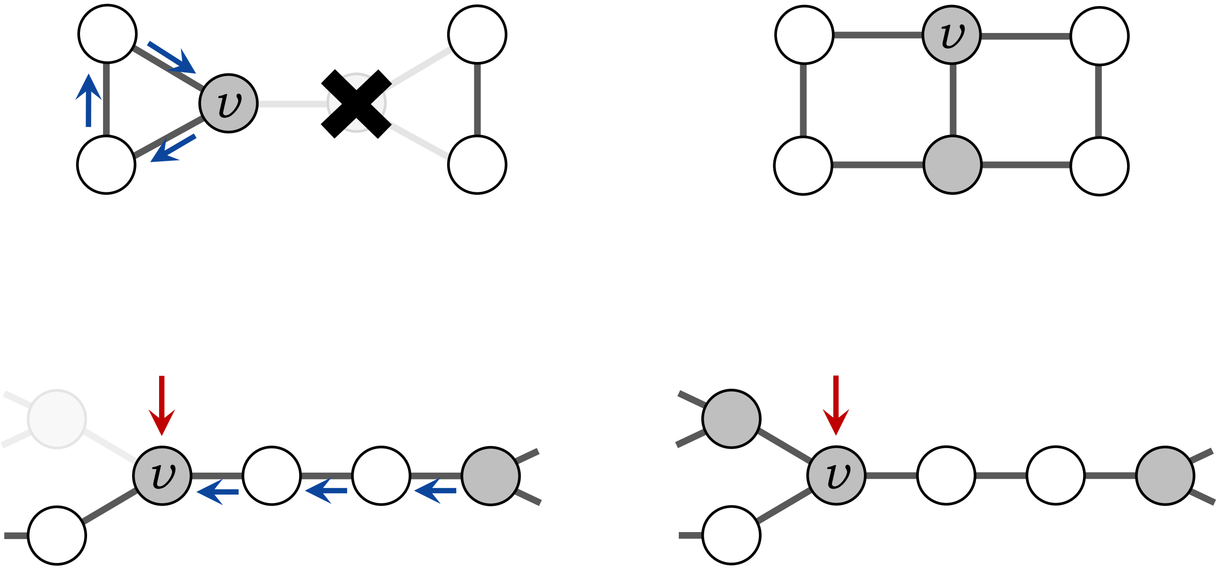
GNNs with Dropouts



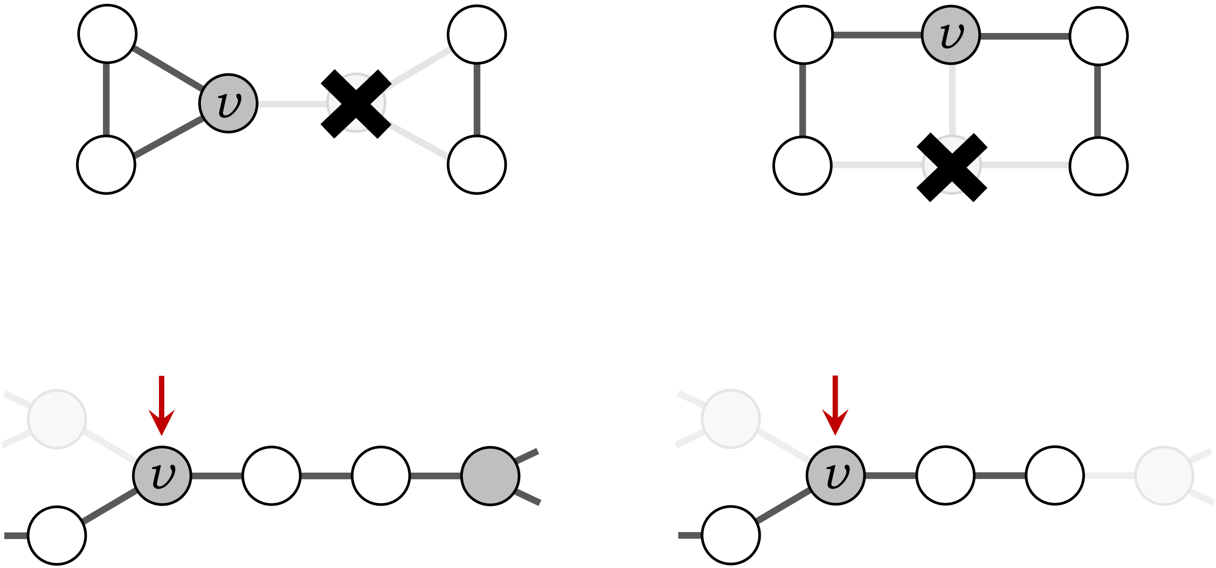
GNNs with Dropouts



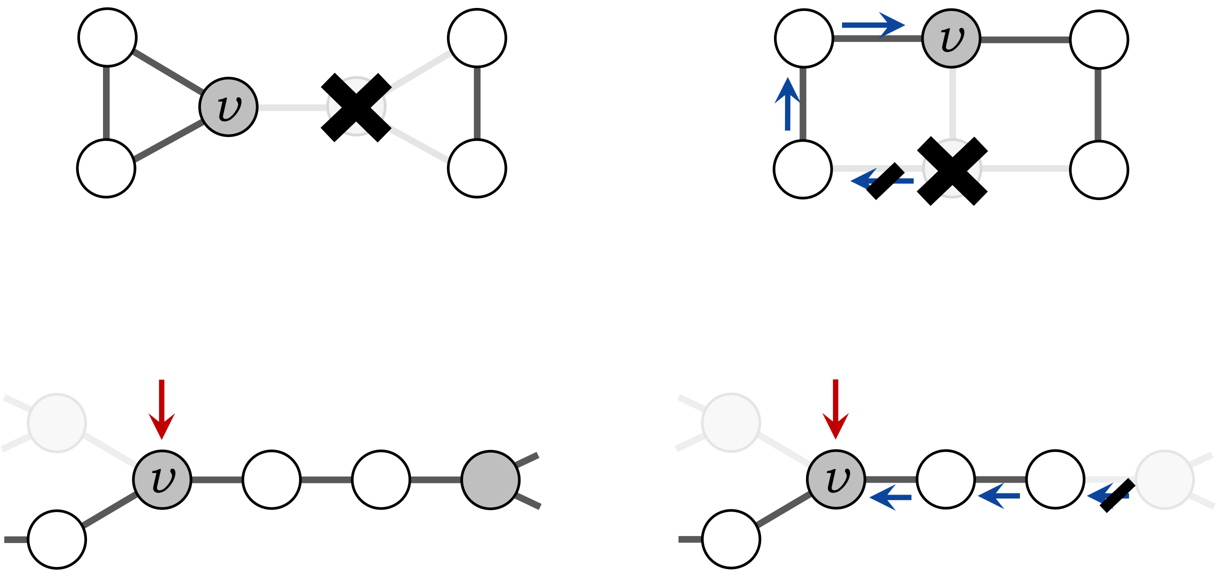
GNNs with Dropouts



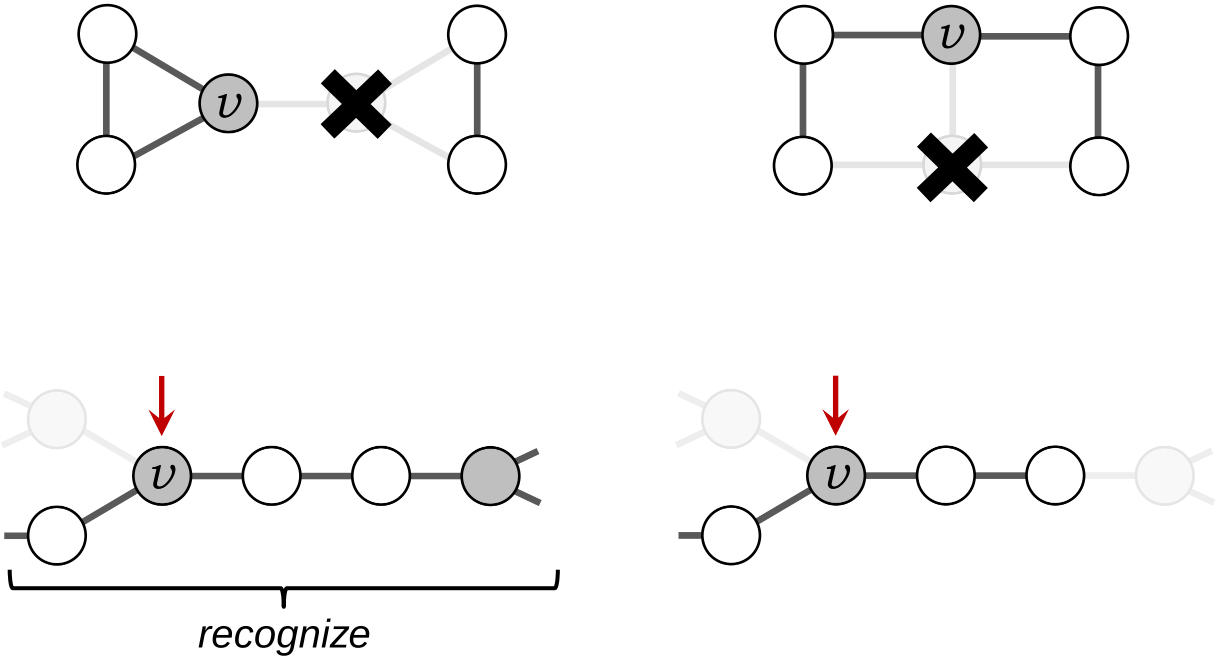
GNNs with Dropouts



GNNs with Dropouts



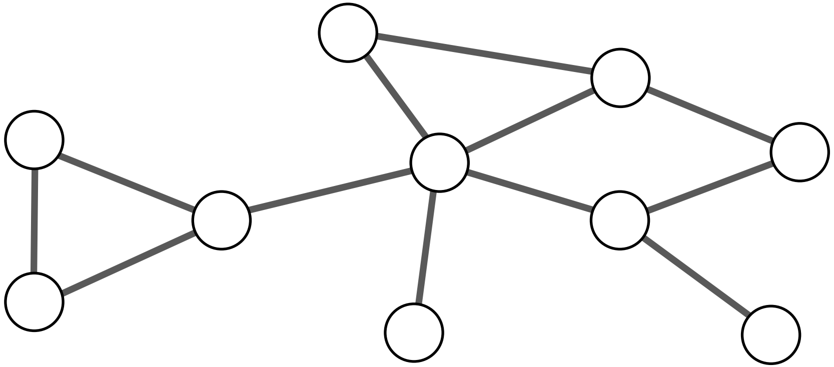
GNNs with Dropouts



GNNs with Dropouts

Multiple runs of the GNN

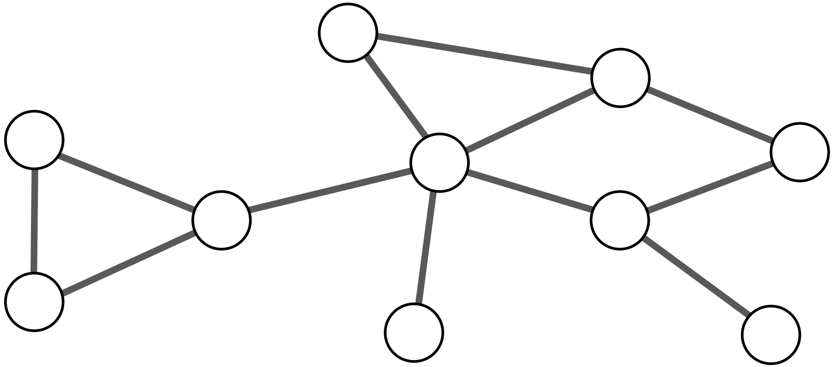
Each node removed with probability *p* independently



GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

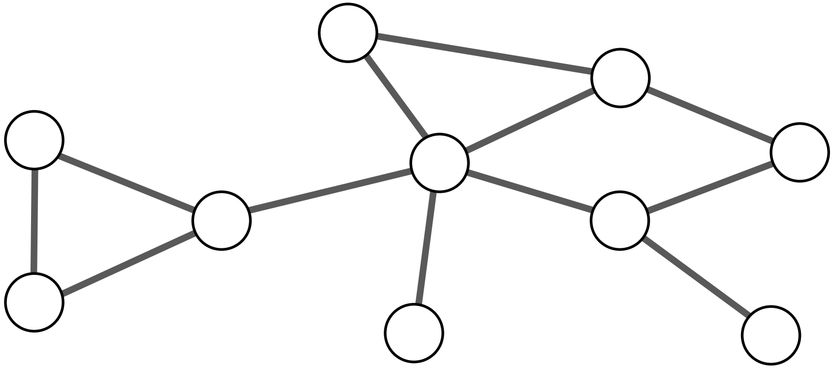


$$h_v = \textsc{RunAggregate}\,(h_v^{[1]}, h_v^{[2]}, \ldots, h_v^{[r]})$$

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability p independently

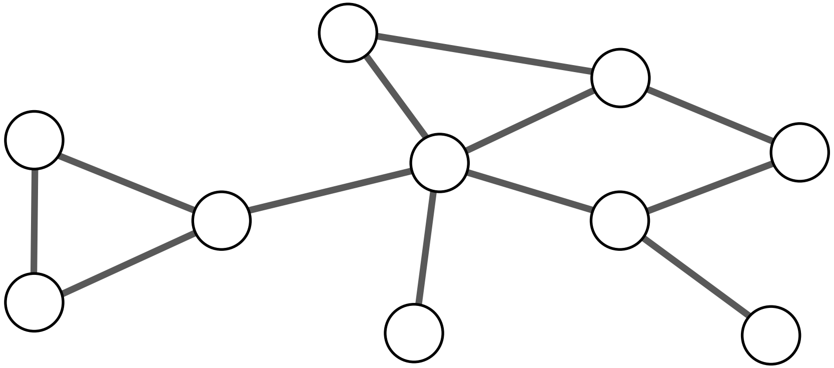


$$h_v = \textsc{RunAggregate}\left(\boxed{h_v^{[1]}}, h_v^{[2]}, \ldots, h_v^{[r]}\right)$$

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

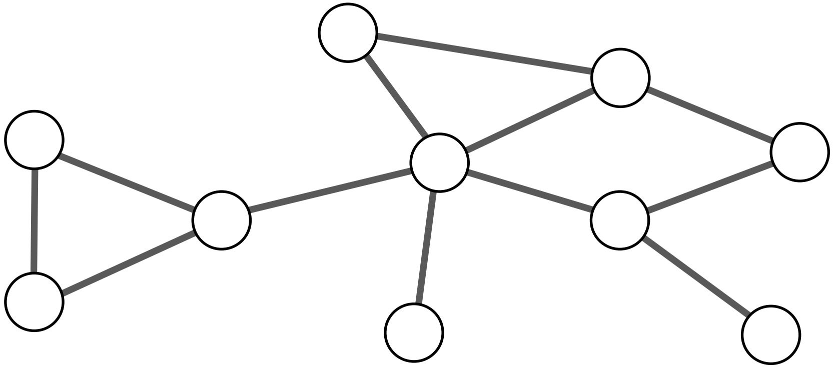


$$h_v = \textsc{RunAggregate}\,(h_v^{[1]}, \boxed{h_v^{[2]}}, \dots, h_v^{[r]})$$

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently



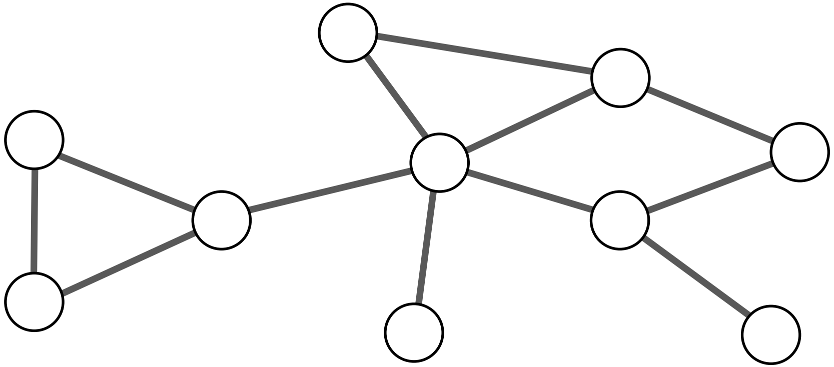
$$h_v = \textsc{RunAggregate}\,(h_v^{[1]}, h_v^{[2]}, \ldots, h_v^{[r]})$$

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

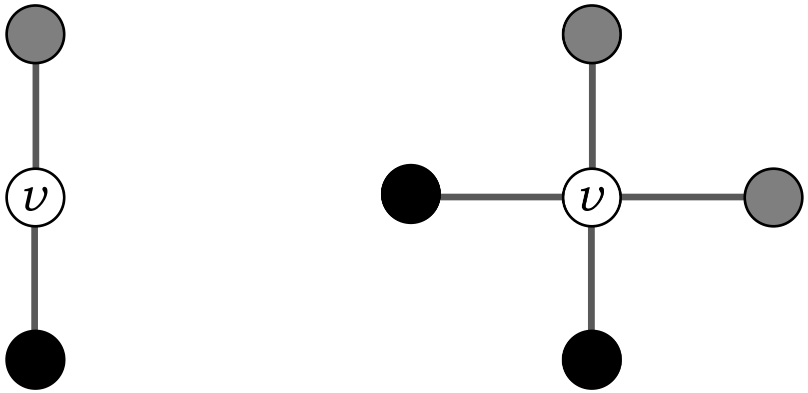
both training and testing!



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

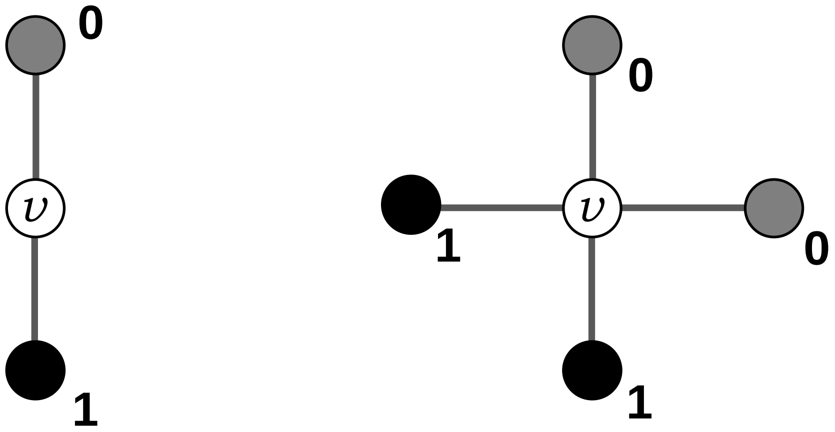
GNNs with Dropouts

MEAN aggregation of neighbors



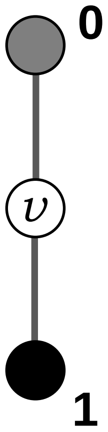
GNNs with Dropouts

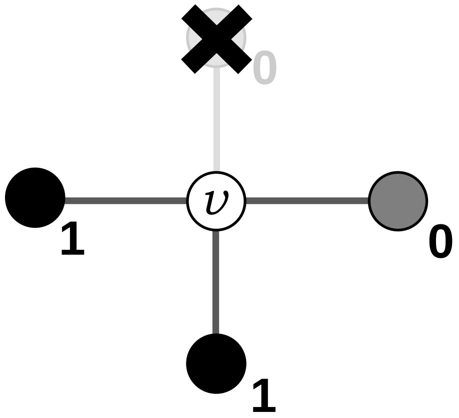
MEAN aggregation of neighbors



GNNs with Dropouts

MEAN aggregation of neighbors

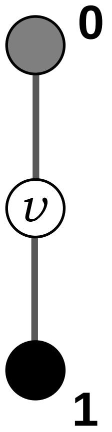




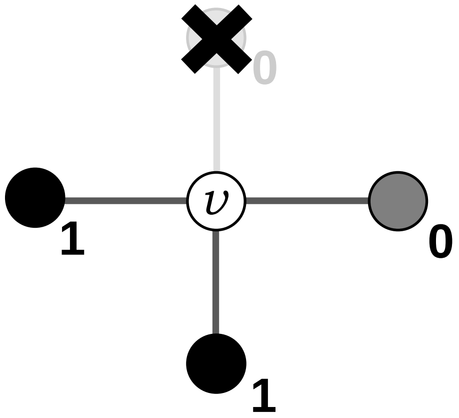
MEAN = 0.66

GNNs with Dropouts

MEAN aggregation of neighbors



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



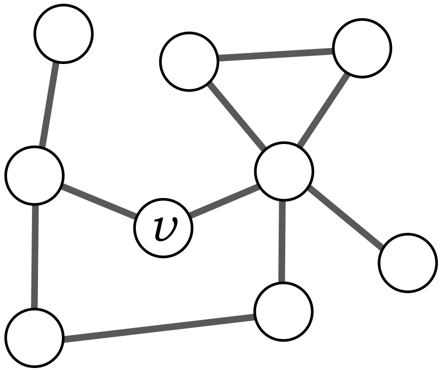
MEAN = 0.66

DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

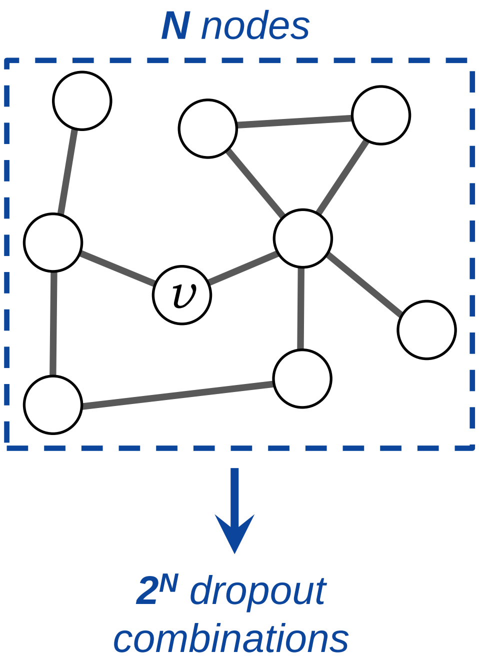


DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead



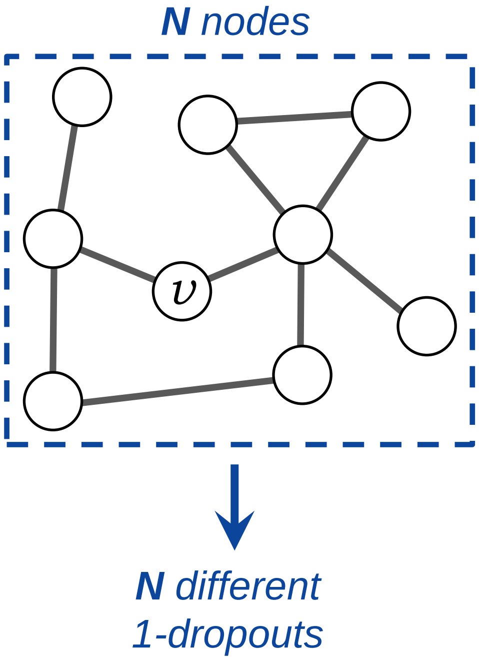
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



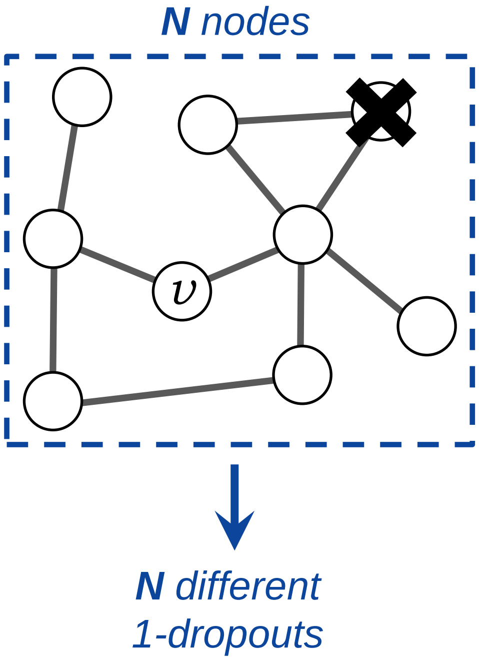
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



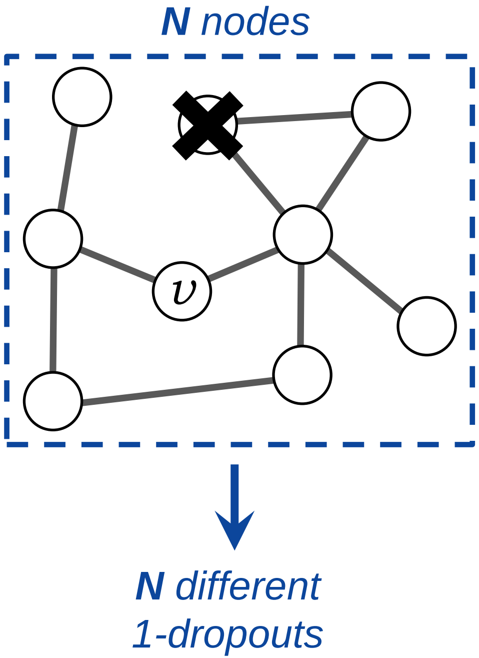
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



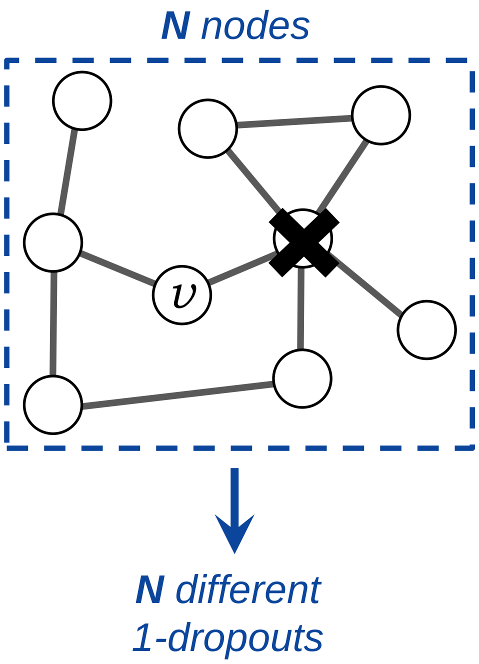
DropGNN with 1-dropouts

More runs:

\+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



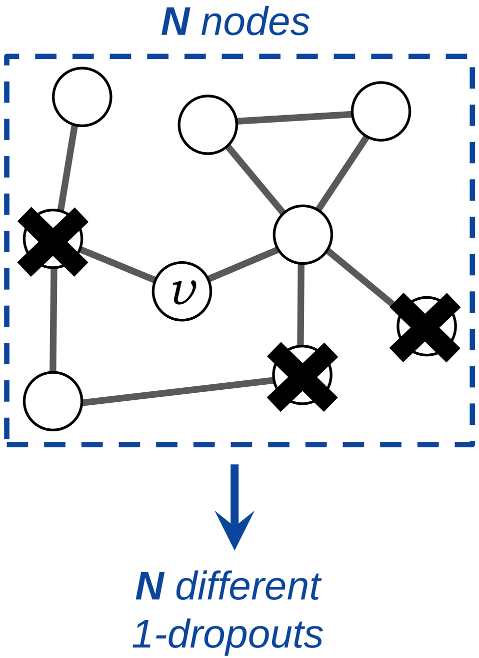
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



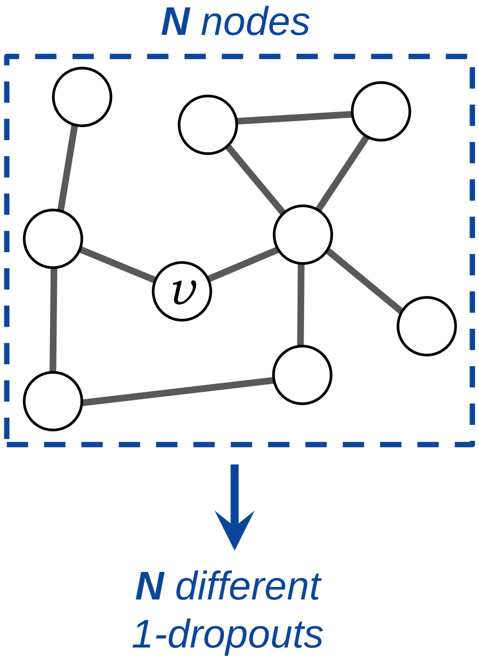
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



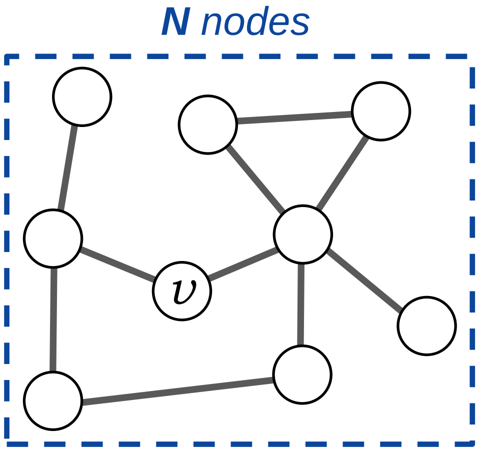
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

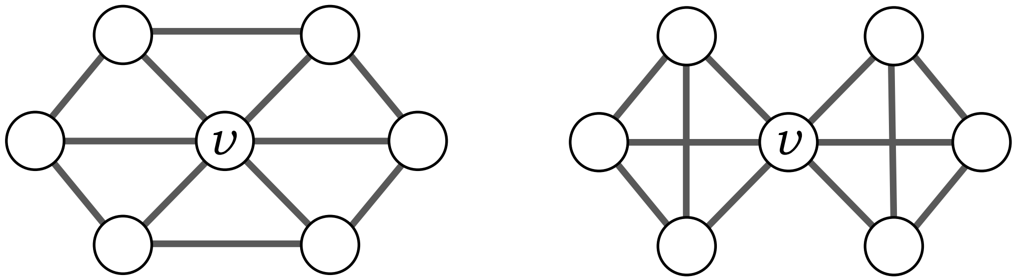
Observe every *1-dropout*



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

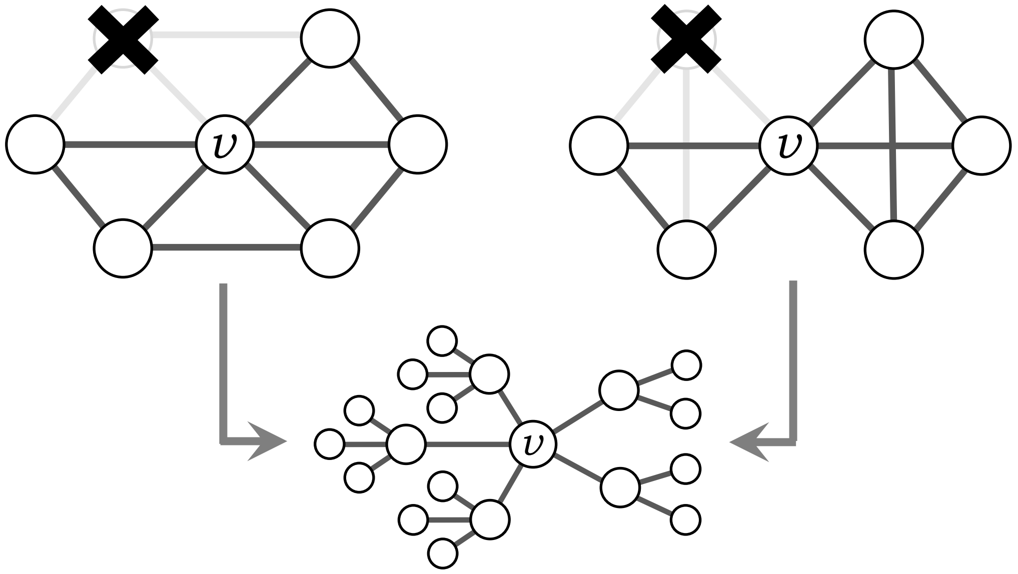
DropGNN with 1-dropouts

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



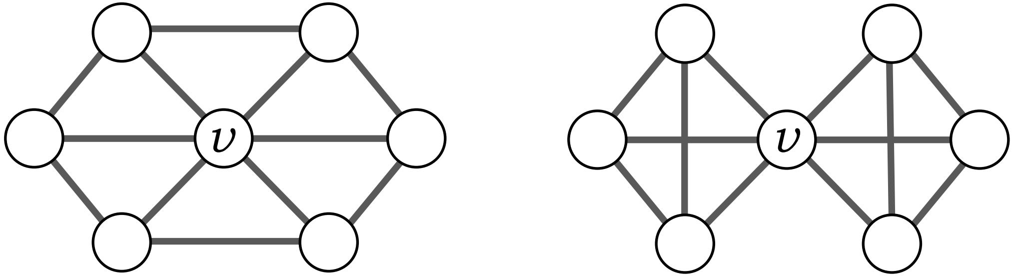
DropGNN with 1-dropouts

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



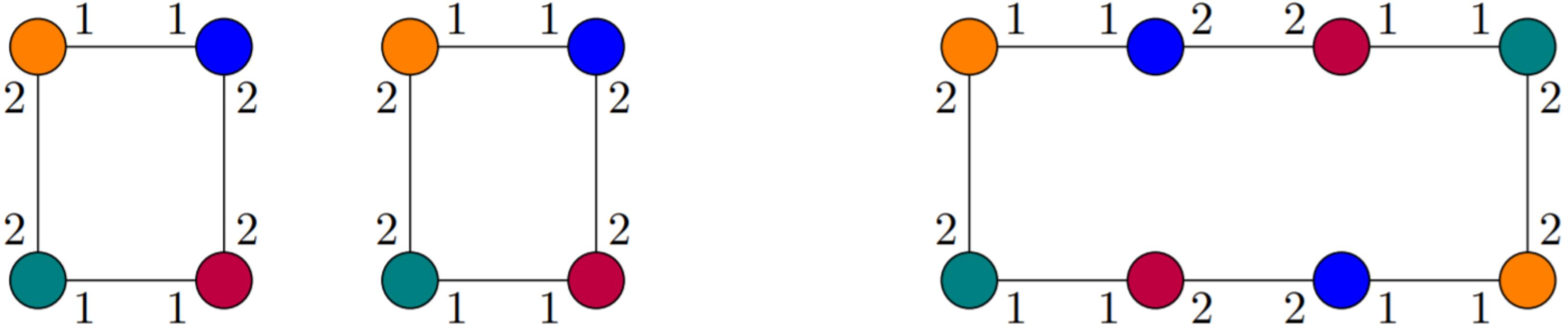
DropGNN with 1-dropouts

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

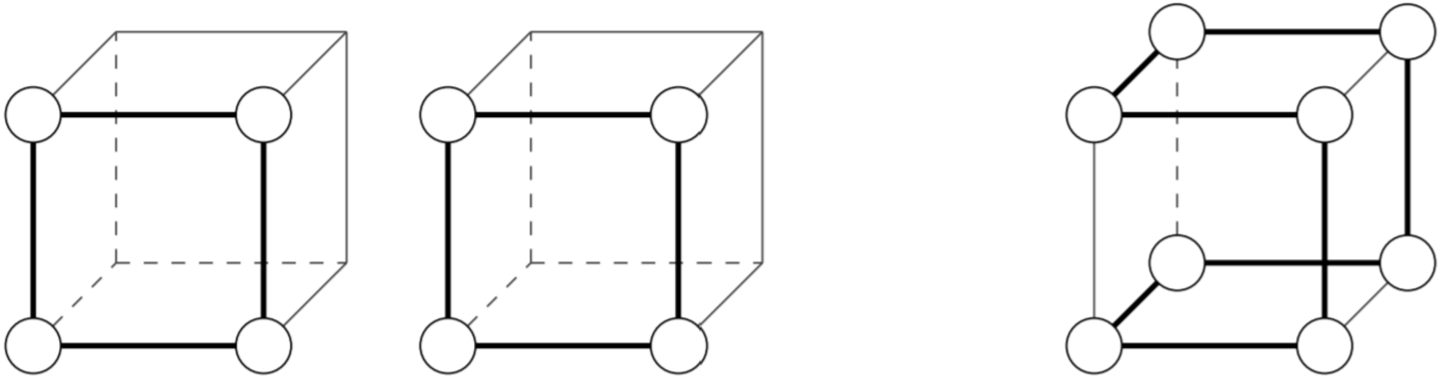


Theorem: in DropGNNs *with port numbers,* any two graphs can be distinguished with 1-dropouts.

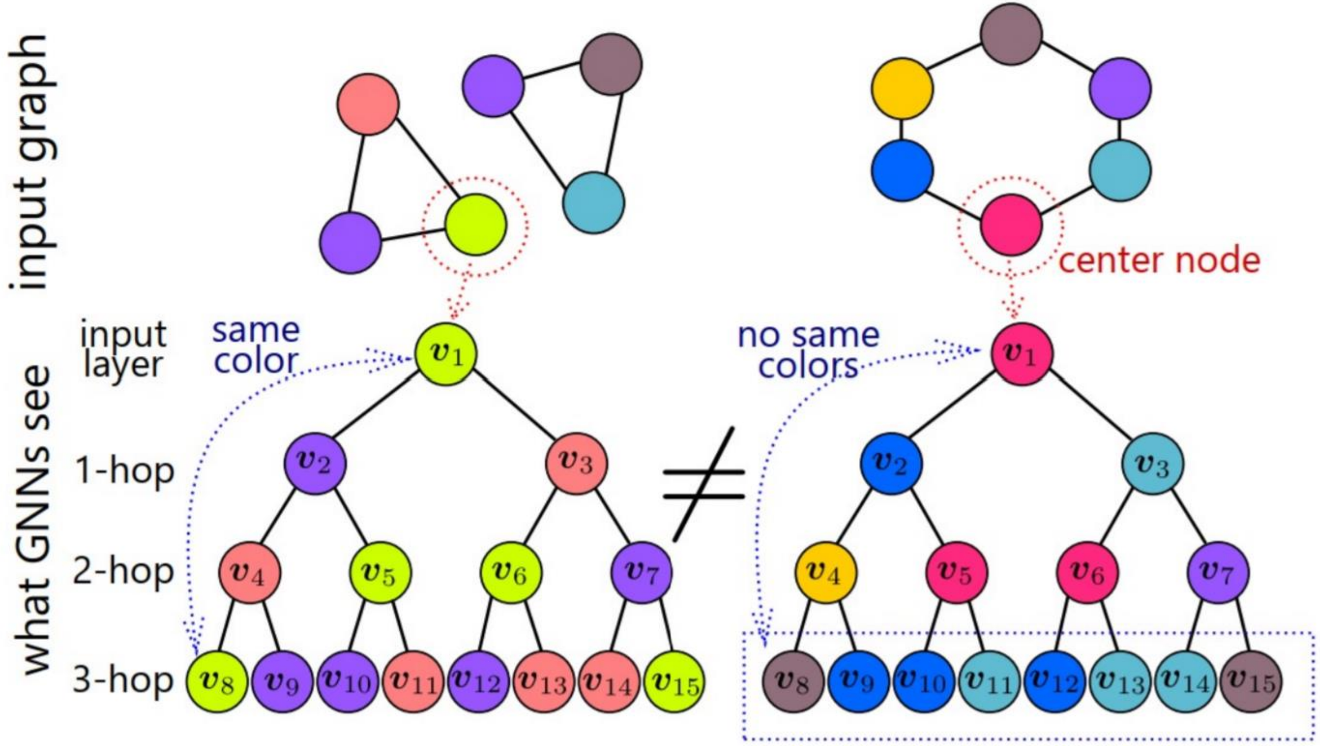
Port Numbers



Angle Features

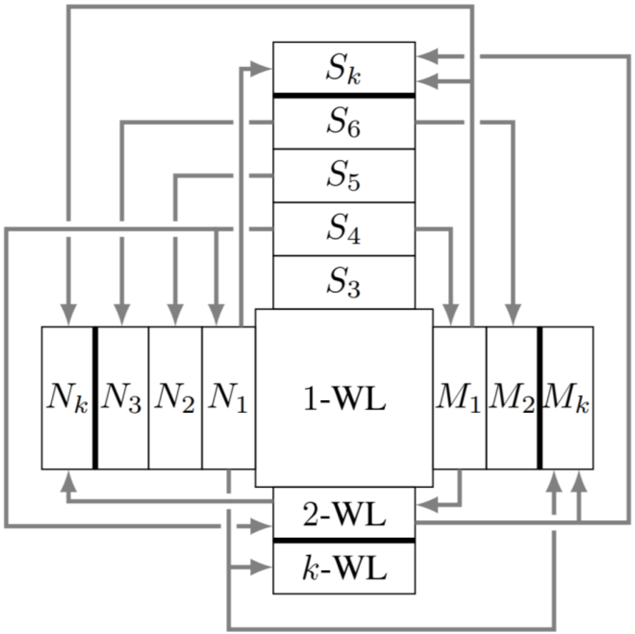


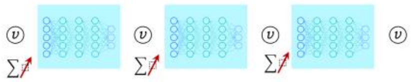
Random Features



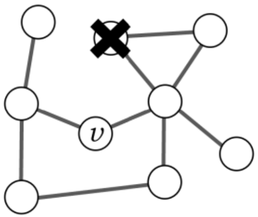
A Theoretical Comparison of Graph Neural Network Extensions

Pál András Papp [1] Roger Wattenhofer [1]

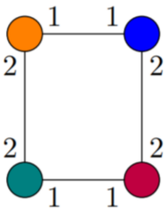




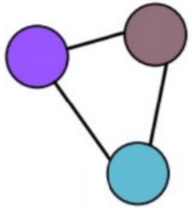
Base GNN



DropGNN

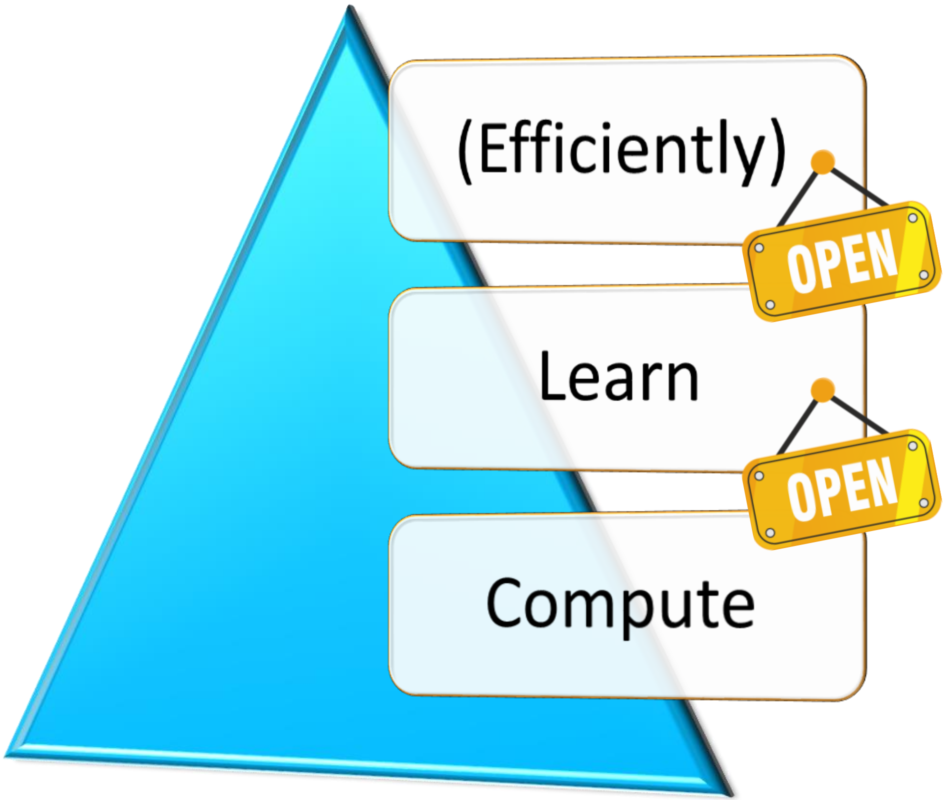


Ports



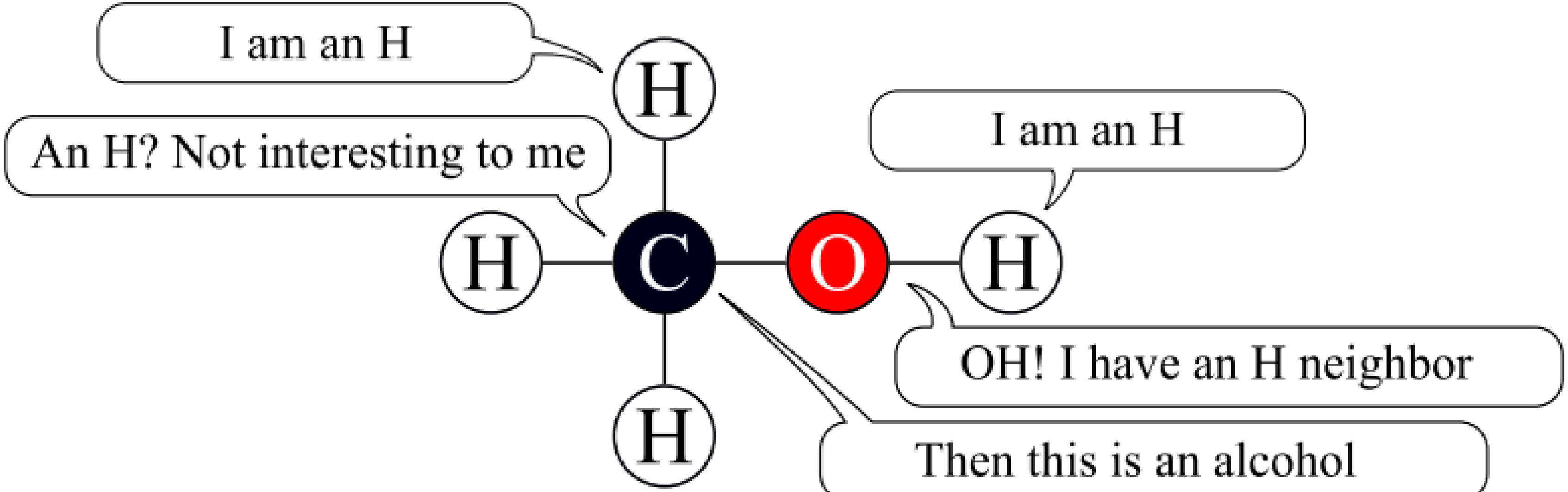
Rand IDs

Reminiscent of Advice Complexity?


(Efficiently)
OPEN
Learn
OPEN
Compute

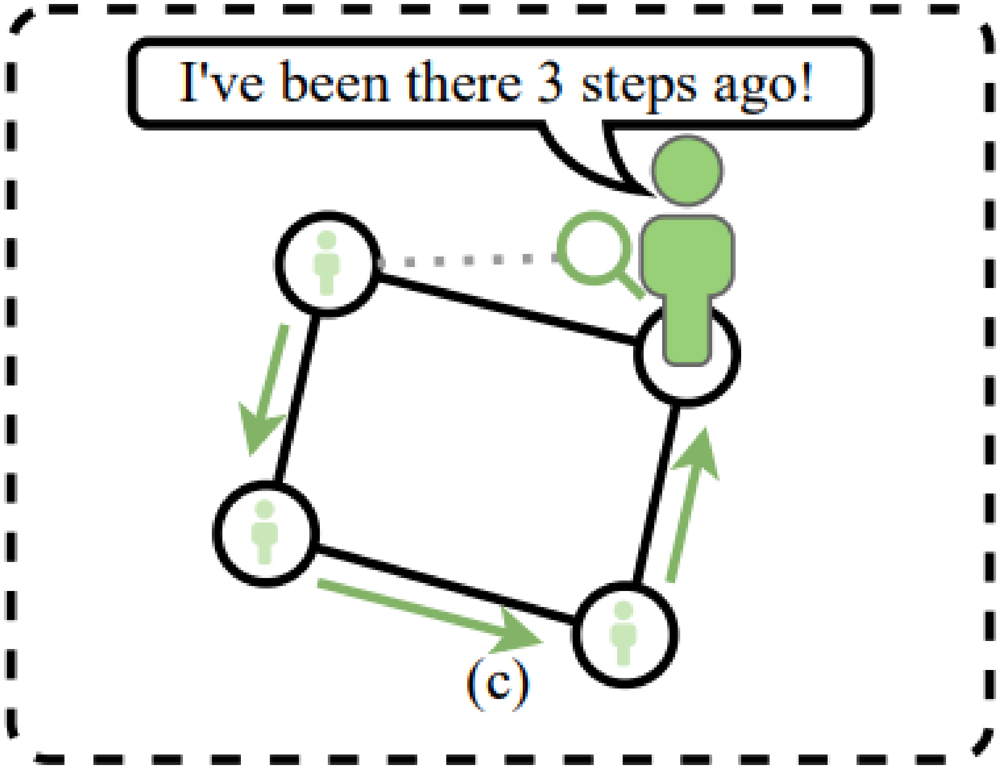
Without Aggregation?

Asynchronous Neural Networks for Learning in Graphs



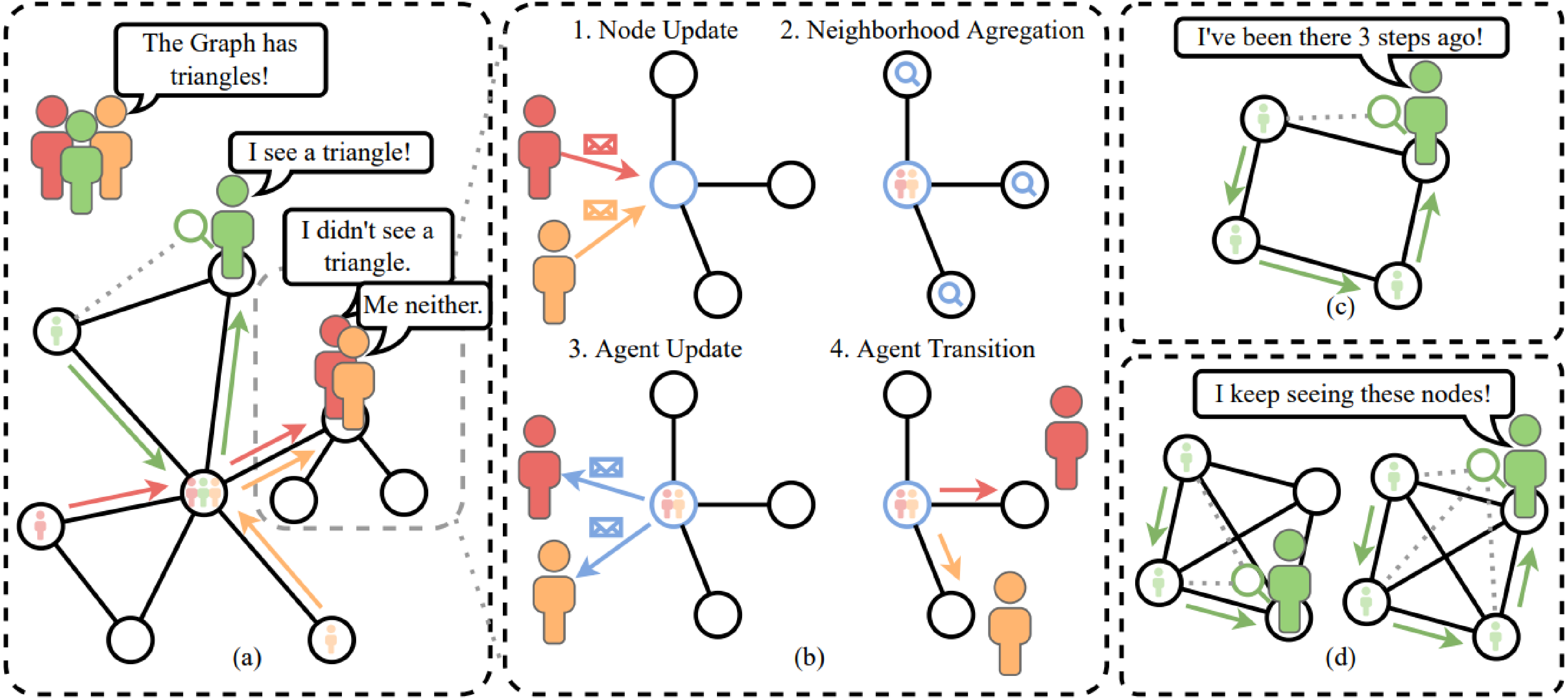
AGENT-BASED GRAPH NEURAL NETWORKS

Karolis Martinkus[1], **Pál András Papp**[2], **Benedikt Schesch**[1], **Roger Wattenhofer**[1]
[1]ETH Zurich [2]Computing Systems Lab, Huawei Zurich Research Center



Agent-based Graph Neural Networks

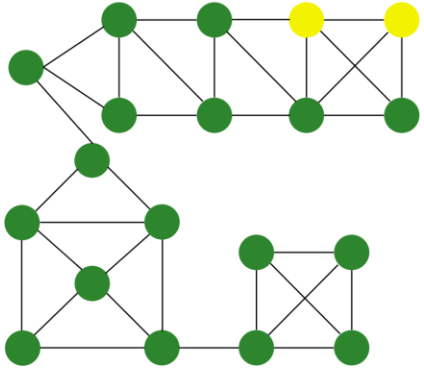
Karolis Martinkus[1], Pál András Papp[2], Benedikt Schesch[1], Roger Wattenhofer[1]



Model	4-CYCLES [59]	CIRCULAR SKIP LINKS [15]	2-WL
GIN [75]	50.0 ±0.0	10.0 ±0.0	50.0 ±0.0
GIN with random features [64; 1]	99.7 ±0.4	95.8 ±2.1	92.4 ±1.6
SMP [71]	**100.0 ±0.0**	**100.0 ±0.0**	50.0 ±0.0
DROPGIN [59]	**100.0 ±0.0**	**100.0 ±0.0**	**100.0 ±0.0**
ESAN [8]	**100.0 ±0.0**	**100.0 ±0.0**	**100.0 ±0.0***
1-2-3 GNN [53]	**100.0 ±0.0**	**100.0 ±0.0**	**100.0 ±0.0**†
PPGN [51]	**100.0 ±0.0**	**100.0 ±0.0**	50.0 ±0.0
CRAWL [67]	**100.0 ±0.0**	**100.0 ±0.0**	**100.0 ±0.0**
RANDOM WALK AGENTNET	**100.0 ±0.0**	**100.0 ±0.0**	50.5 ±4.5
SIMPLIFIED AGENTNET	**100.0 ±0.0**	**100.0 ±0.0**	**100.0 ±0.0**
AGENTNET	**100.0 ±0.0**	**100.0 ±0.0**	**100.0 ±0.0**

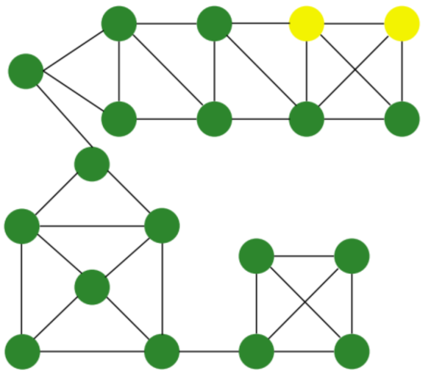
Explainable GNNs

GraphChef: Learning the Recipe of Your Dataset

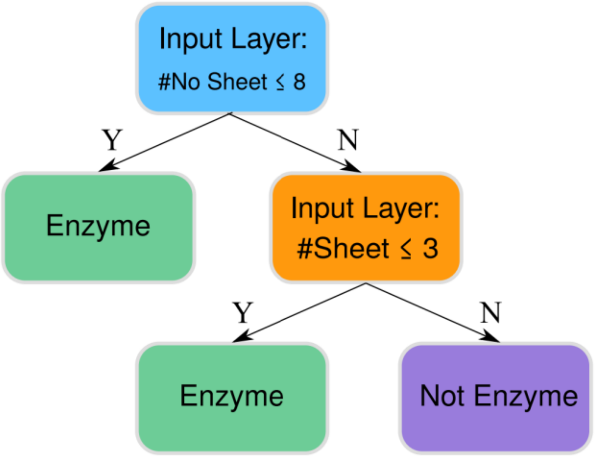


(a)

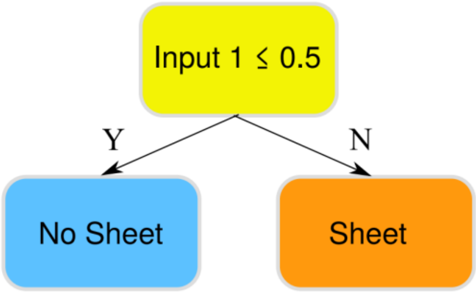
GraphChef: Learning the Recipe of Your Dataset



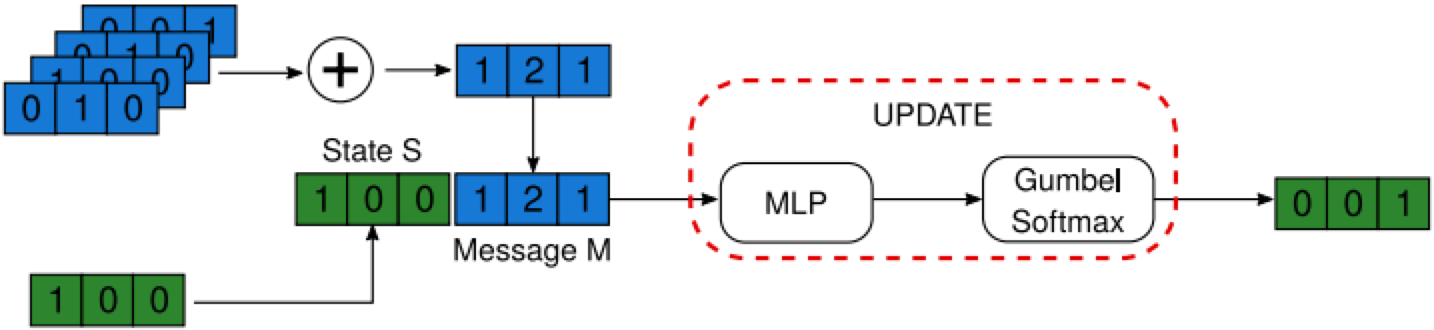
(a)



(b)



(c)



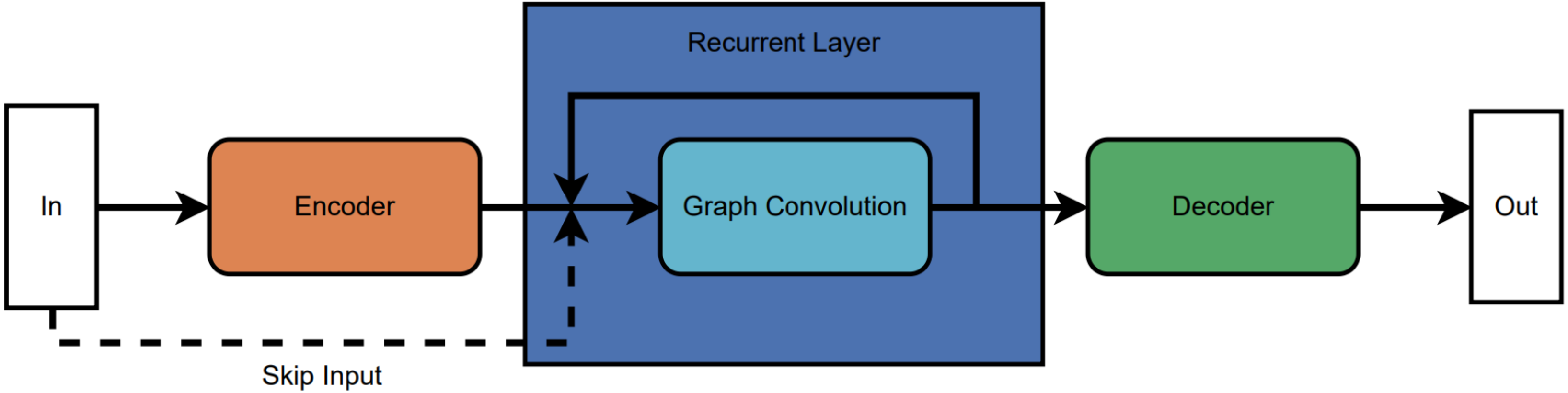
Reminiscent of Stone Age Model?

Extrapolation

Learning Graph Algorithms With Recurrent Graph Neural Networks

Florian Grötschla*,[1] **Joël Mathys***,[1] **Roger Wattenhofer**[1]

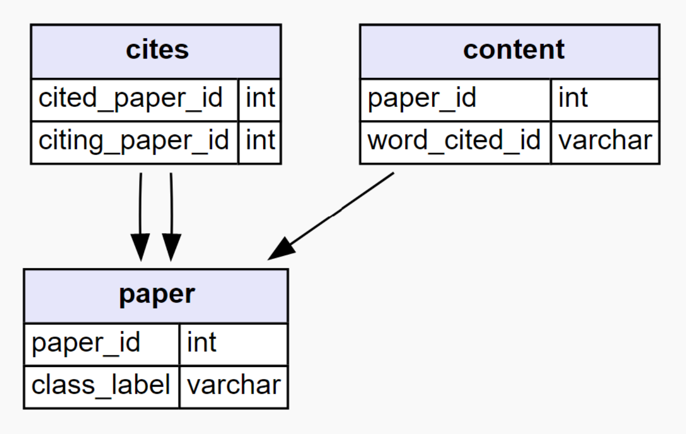
[1] ETH Zurich
fgroetschla@ethz.ch, jmathys@ethz.ch, wattenhofer@ethz.ch

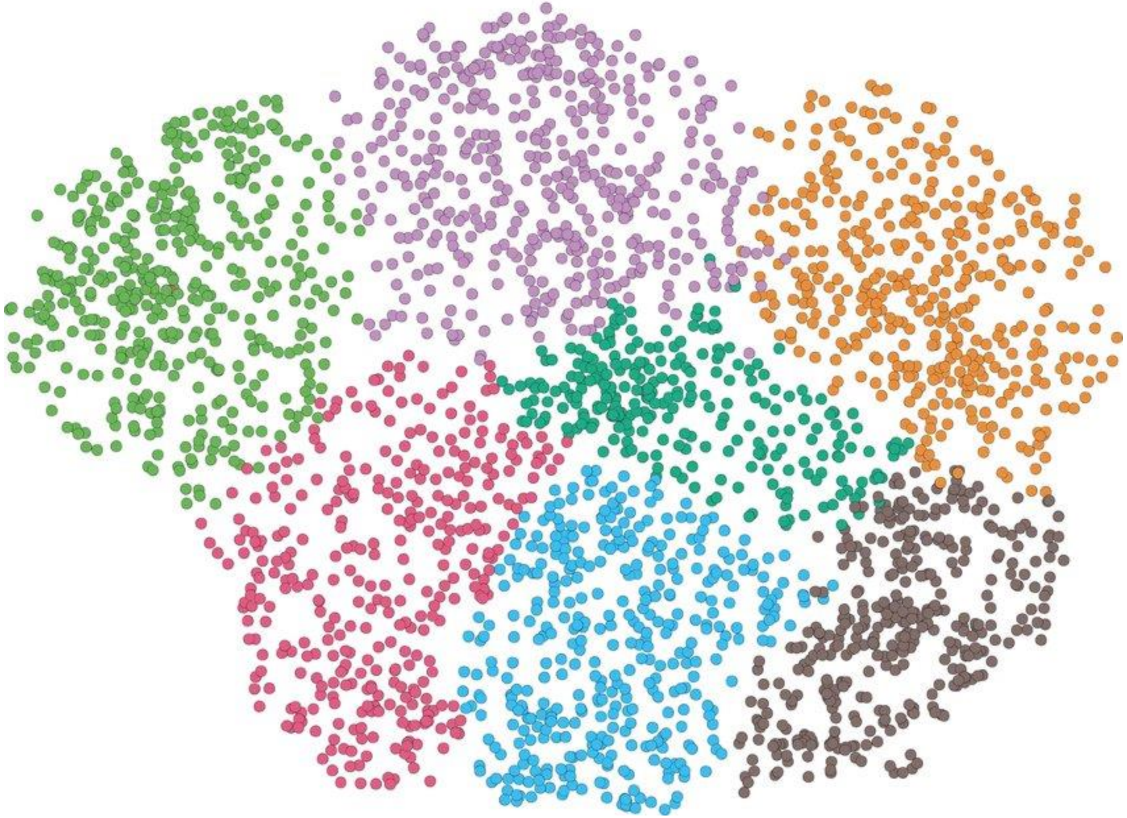


Towards Learning *Algorithms*?

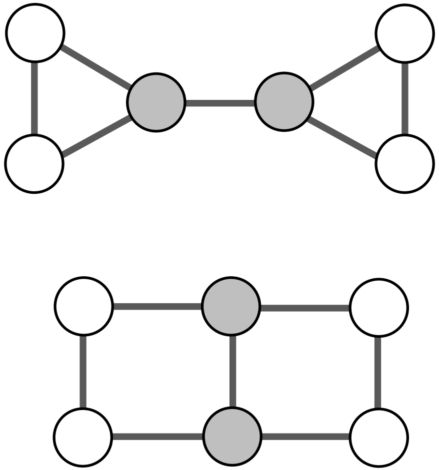
GNN Benchmarks

Example: CORA Benchmark





Example: CORA Benchmark



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

Can Good GNN Benchmarks Exist?

Networks

Social Networks

Neural Networks

Mobile Networks

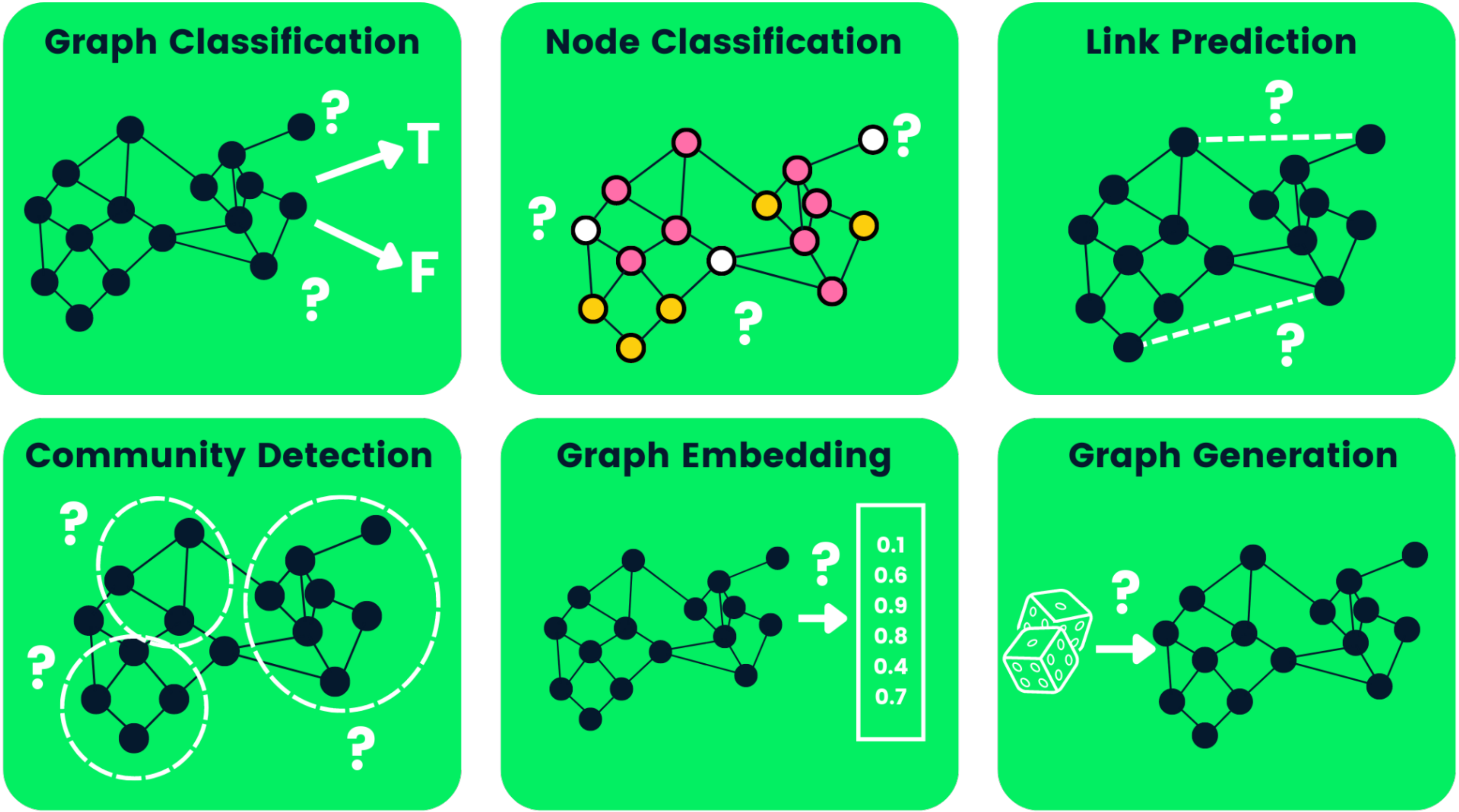
Wireless Networks

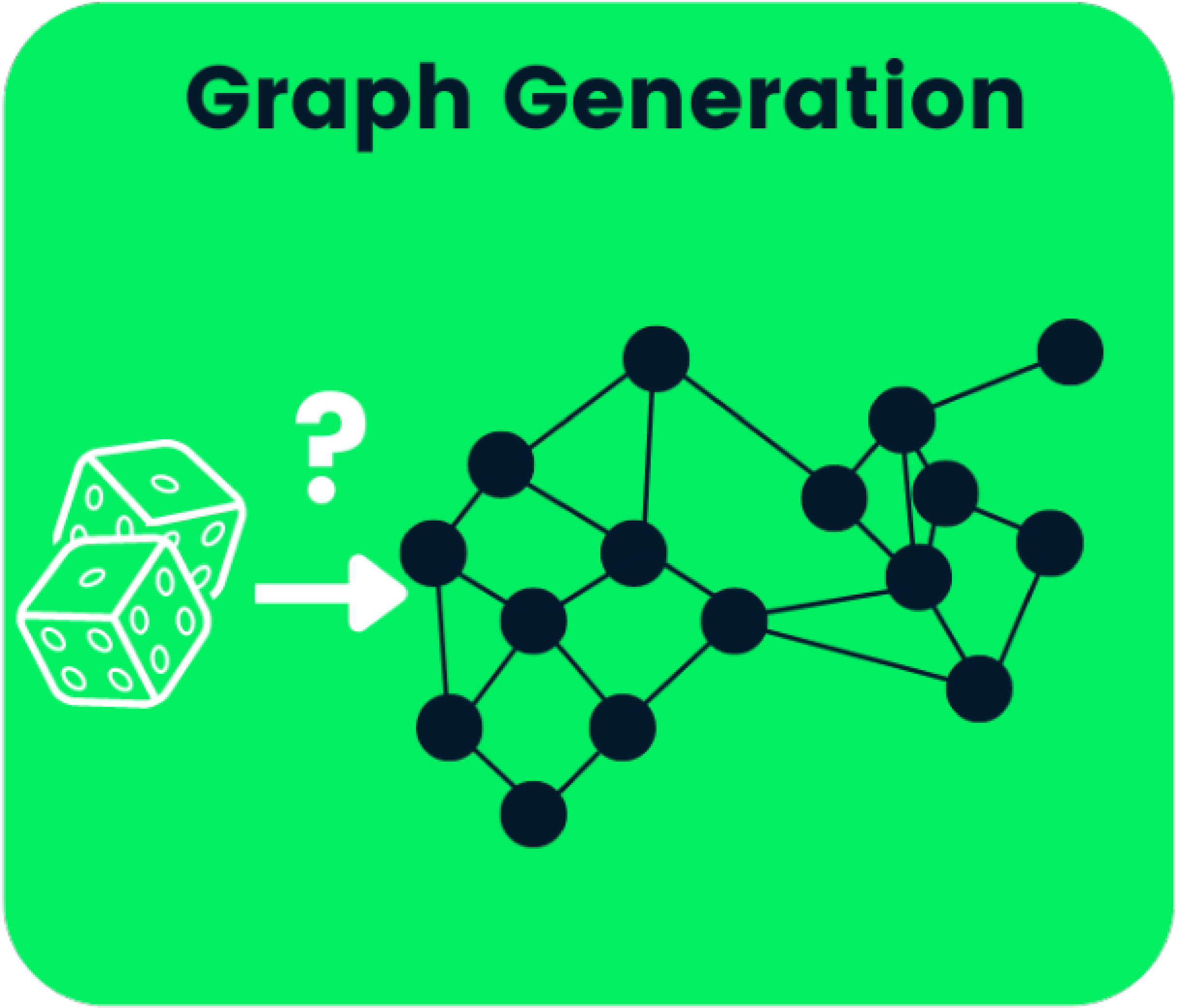
Financial Networks

Economic Networks

Biological Networks

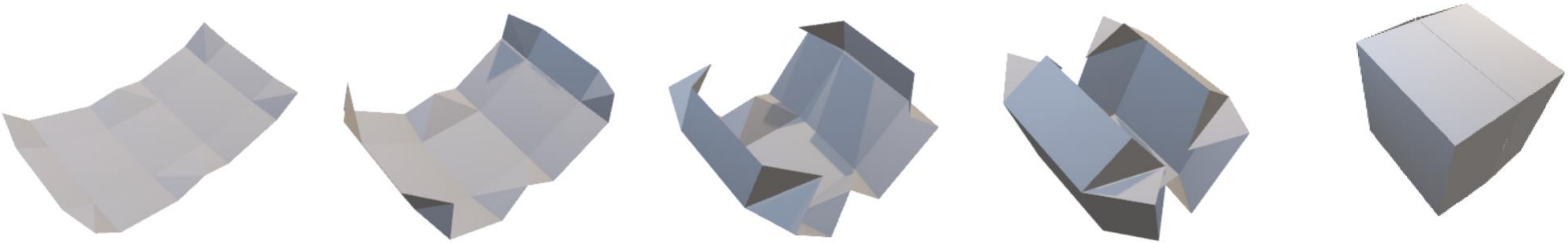
Computer Networks

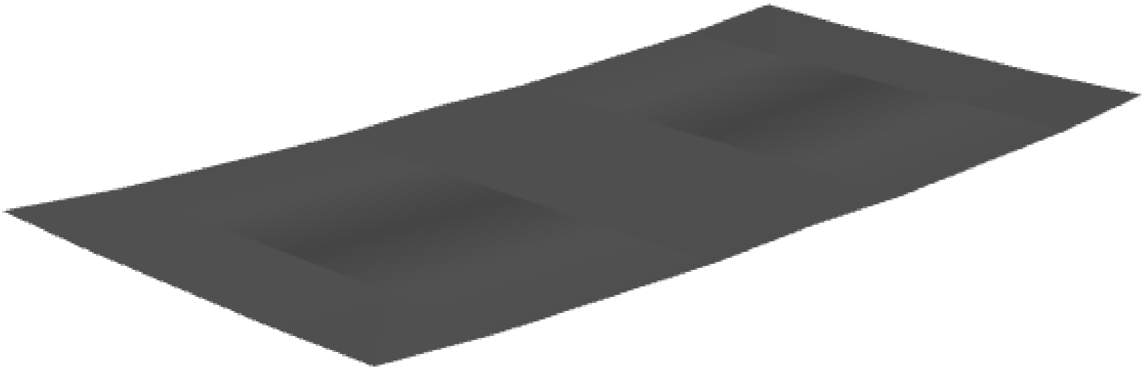

Graph Classification
?
T
F
?
Node Classification
?
?
?
Link Prediction
?
?
Community Detection
?
?
?
Graph Embedding
?
0.1
0.6
0.9
0.8
0.4
0.7
Graph Generation
?


Graph Generation

Automating Rigid Origami Design

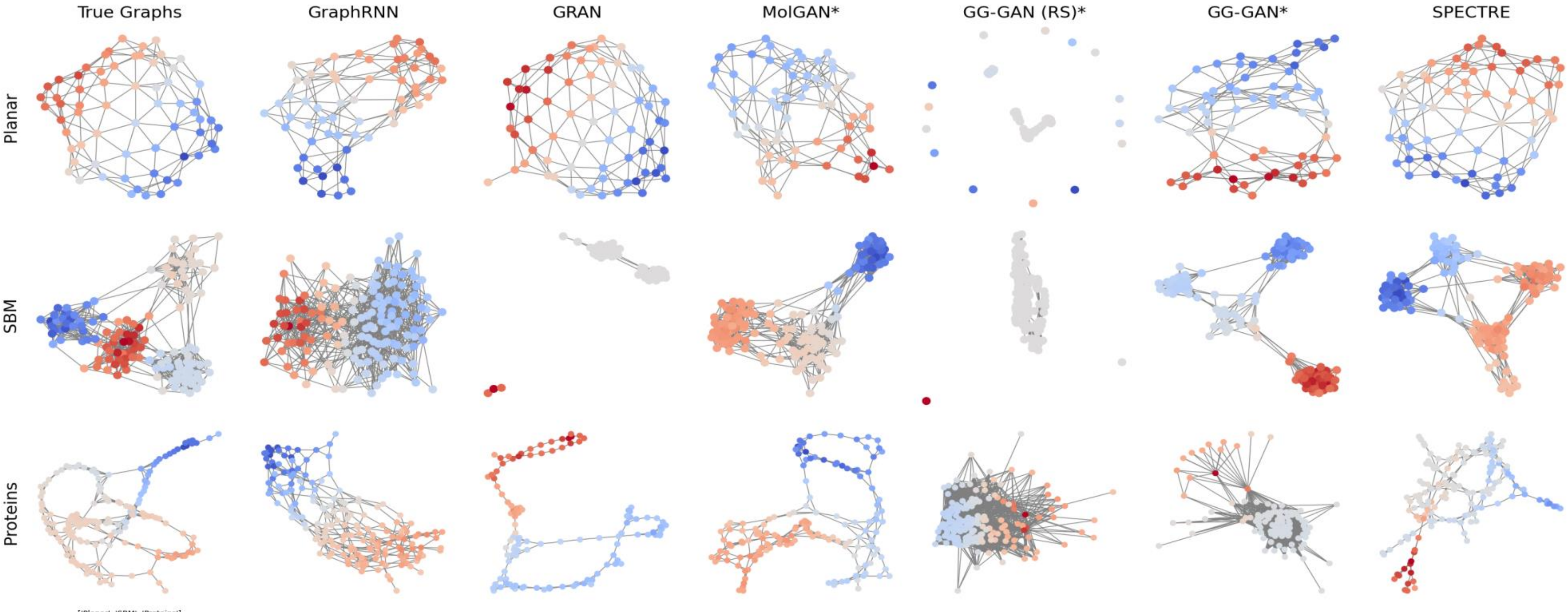
Jeremia Geiger, Karolis Martinkus, Oliver Richter, Roger Wattenhofer



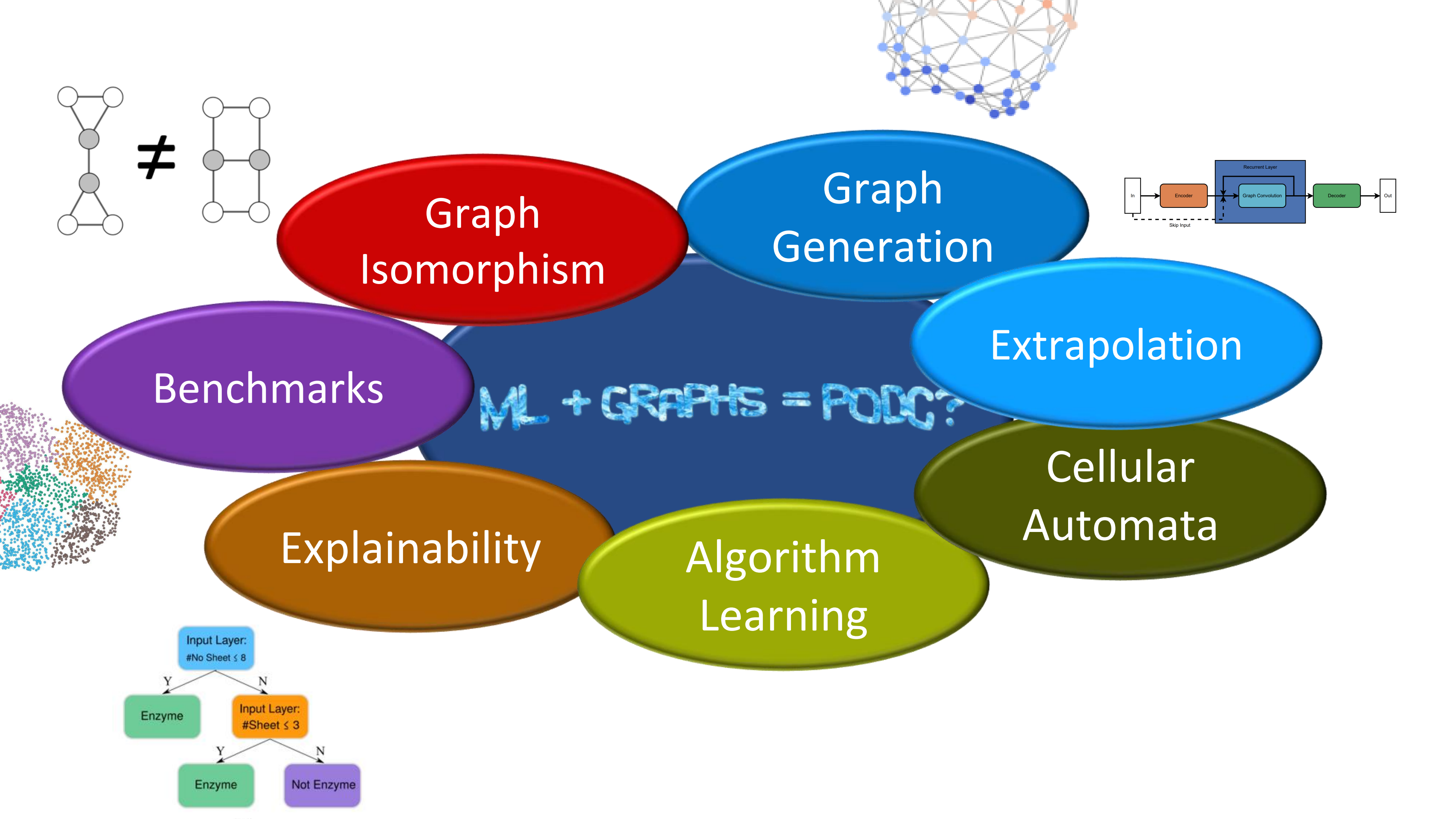


SPECTRE : Spectral Conditioning Helps to Overcome the Expressivity Limits of One-shot Graph Generators

Karolis Martinkus [1] **Andreas Loukas** [* 2] **Nathanaël Perraudin** [* 3] **Roger Wattenhofer** [1]




120"
100"
85"
The Bigger Picture


≠
Graph Isomorphism
Graph Generation
Recurrent Layer
In
Encoder
Graph Convolution
Decoder
Out
Skip Input
Benchmarks
ML + GRAPHS = PODC?
Extrapolation
Cellular Automata
Explainability
Algorithm Learning
Input Layer: #No Sheet ≤ 8
Y
N
Enzyme
Input Layer: #Sheet ≤ 3
Y
N
Enzyme
Not Enzyme

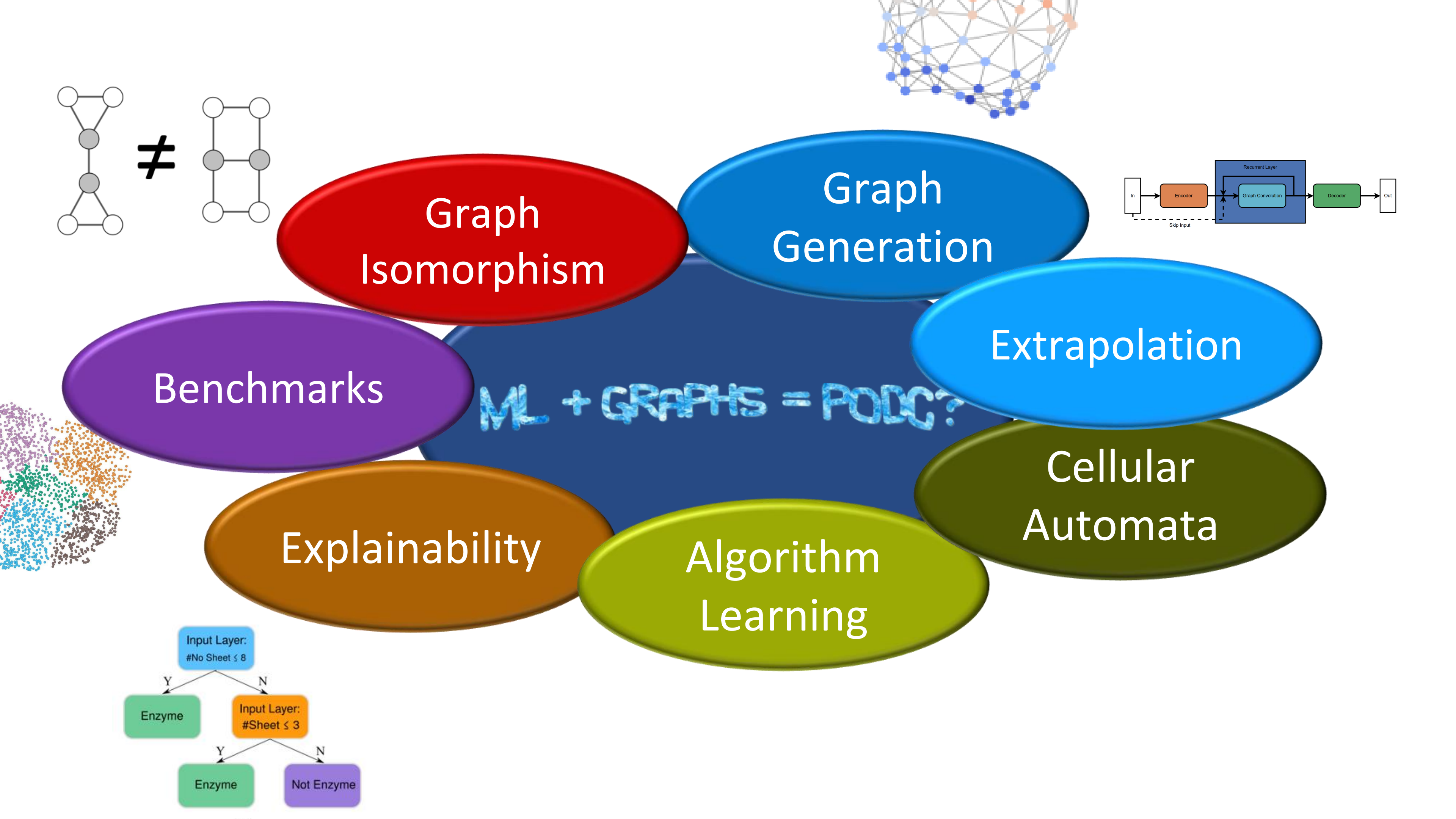
Distributed Computing (DC)

Machine Learning (ML)


DC

Thank You!

Questions & Comments?



Graph Isomorphism
Graph Generation
Recurrent Layer
In
Encoder
Graph Convolution
Decoder
Out
Skip Input
Benchmarks
ML + GRAPHS = PODC?
Extrapolation
Cellular Automata
Explainability
Algorithm Learning
Input Layer:
#No Sheet ≤ 8
Y
N
Enzyme
Input Layer:
#Sheet ≤ 3
Y
N
Enzyme
Not Enzyme