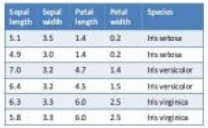
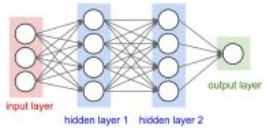
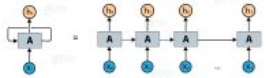
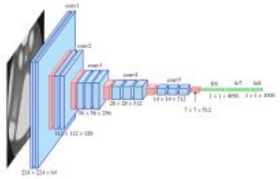
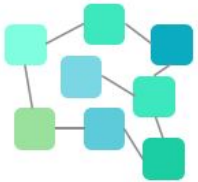
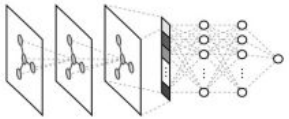
Graph neural networks

Meysam Madadi

Slides credit: Lulia Duta and Andrei Nicolicioiu

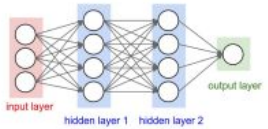
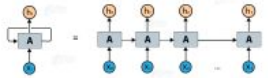
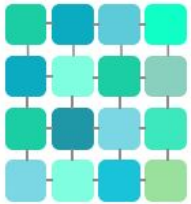
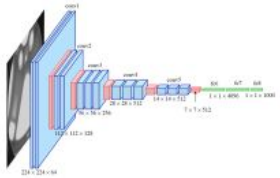
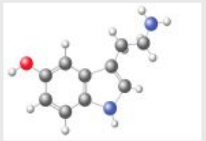
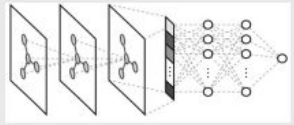
Choose your model

Bitdefender

UNSTRUCTURED				MLP
SEQUENTIAL		 Have a nice day! :)		RNN
GRID				CNN
RELATIONAL STRUCTURE				GNN

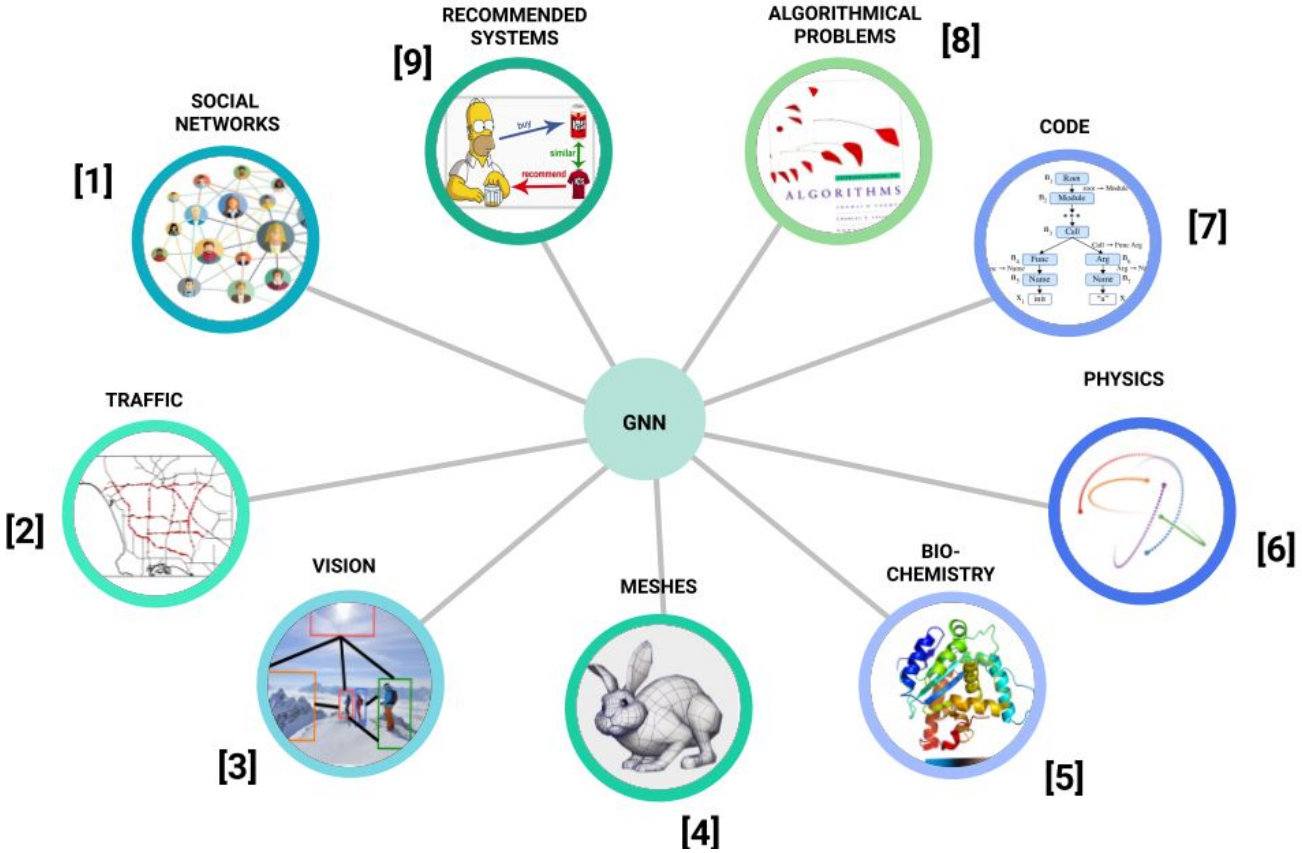
Choose your model

Bitdefender

UNSTRUCTURED				MLP
SEQUENTIAL		 Have a nice day! :)		RNN
GRID				CNN
RELATIONAL STRUCTURE				GNN

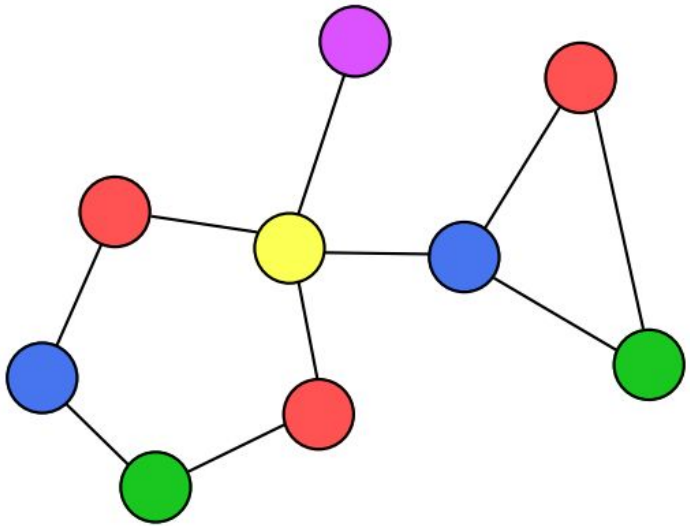
Tasks

Bitdefender



Data: Graph Structure

Bitdefender



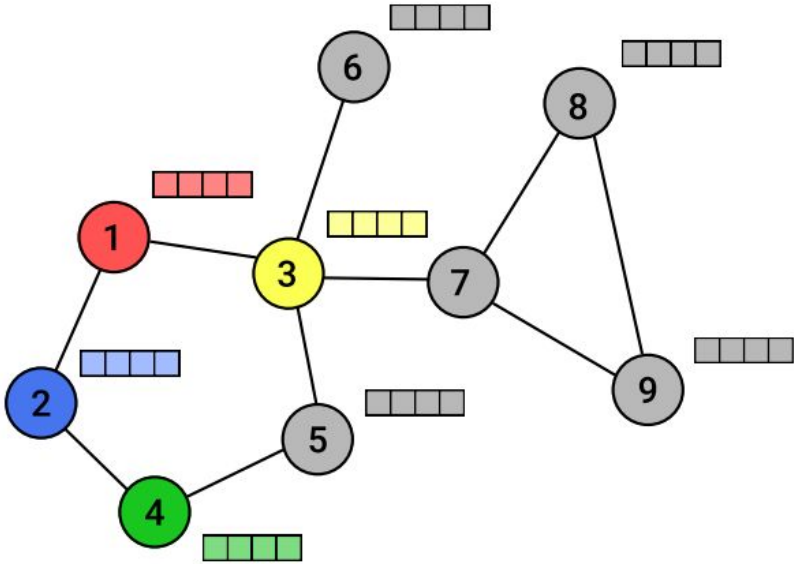
Tasks where we have access or we can create a graph structure.

A graph G is characterized by:

- a set of **nodes**
  $X = \{x_i | i \in 1..N\}$
- connected by **edges**
  $\mathcal{E} = \{e_{ij} | i, j \in 1..N\}$

Data: Graph Structure

Bitdefender



Tasks where we have access or we can create a graph structure.

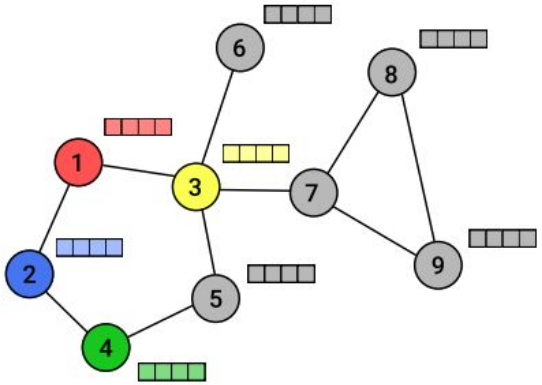
A graph G is characterized by:

- a set of **nodes**
  $X = \{x_i | i \in 1..N\}$
- connected by **edges**
  $\mathcal{E} = \{e_{ij} | i, j \in 1..N\}$

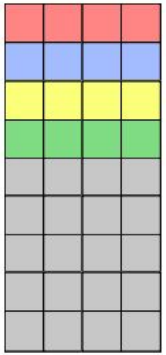
Each node i is characterized by a set of features $x_i \in \mathbb{R}^D$

Data: Graph Structure - Nodes

Bitdefender



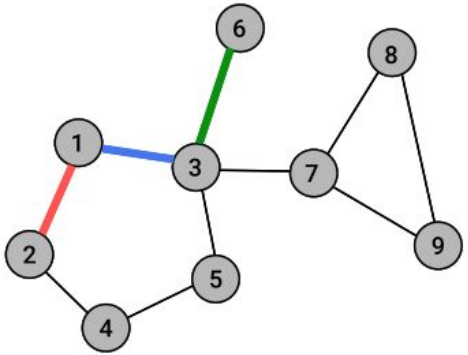
$X \in \mathbb{R}^{N \times D}$



- all the nodes $x_i \in \mathbb{R}^D$ are stacked into a matrix $X \in \mathbb{R}^{N \times D}$
- each row corresponds to a node $x_i \in \mathbb{R}^D$

Data: Graph Structure - Edges

Bitdefender



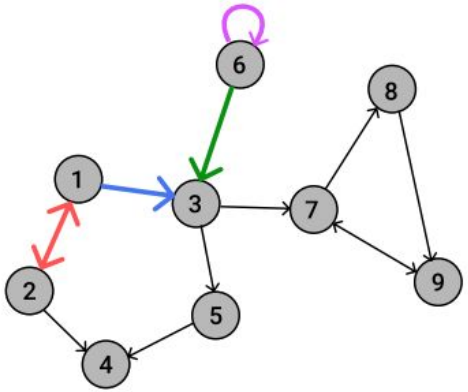
$$A \in \mathbb{R}^{N \times N}$$

	1	2	3	4	5	6	7	8	9
1	0	1	1	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0
3	1	0	0	0	1	1	1	0	0
4	0	1	0	0	1	0	0	0	0
5	0	0	1	1	0	0	0	0	0
6	0	0	1	0	0	0	0	0	0
7	0	0	1	0	0	0	0	1	1
8	0	0	0	0	0	0	1	0	1
9	0	0	0	0	0	0	1	1	0

- the edges $\mathcal{E}$ could be represented by an adjacency matrix $A \in \mathbb{R}^{N \times N}$
- $a_{ij} \neq 0$ if there is an edge between node i and node j

Data: Graph Structure - Edges

Bitdefender



$$A \in \mathbb{R}^{N \times N}$$

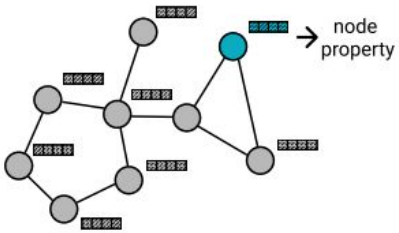
	1	2	3	4	5	6	7	8	9
1	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0
3	1	0	0	0	0	1	0	0	0
4	0	1	0	0	1	0	0	0	0
5	0	0	1	1	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	0	1
8	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	1	1	0

- un-directed graph: adjacency matrix is symmetric
- directed graph: adjacency matrix is **not** symmetric
- $a_{ij} \neq 0$ if there is an edge **from j to i**
- a graph could contain *self-loops*

GNNs Goal

Bitdefender

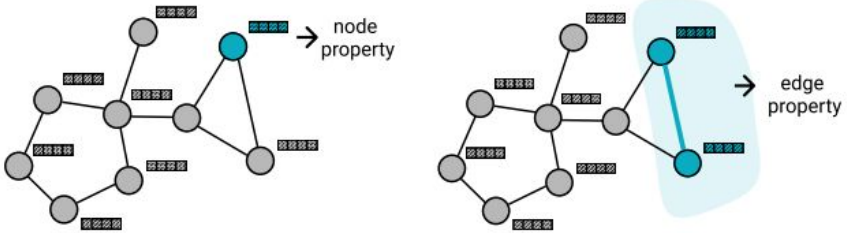
- Based on the node features (X) and the graph structure (A), we want to learn a representation of the graph.
- Depending on the task, the representation could be:
  1. node level: $Y \in \mathbb{R}^{N \times K}$



GNNs Goal

Bitdefender

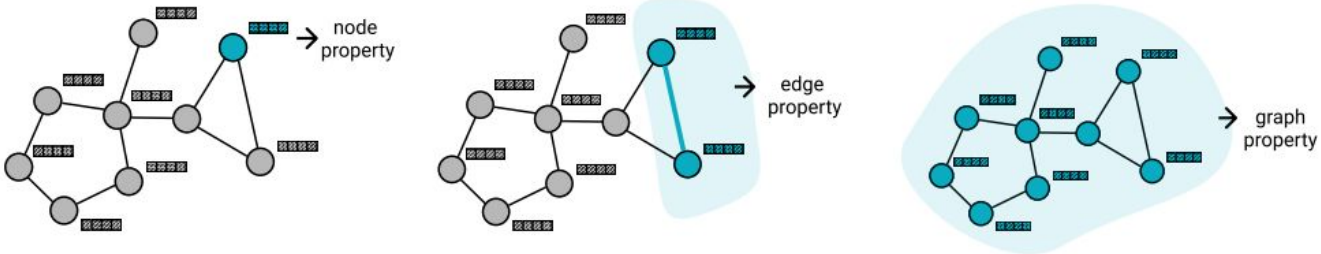
- Based on the node features (X) and the graph structure (A), we want to learn a representation of the graph.
- Depending on the task, the representation could be:
  1. node level: $Y \in \mathbb{R}^{N \times K}$
  2. edge level: $Y \in \mathbb{R}^{M \times K}$



GNNs Goal

Bitdefender

- Based on the node features (X) and the graph structure (A), we want to learn a representation of the graph.
- Depending on the task, the representation could be:
  1. node level: $Y \in \mathbb{R}^{N \times K}$
  2. edge level: $Y \in \mathbb{R}^{M \times K}$
  3. graph level: $Y \in \mathbb{R}^{K}$



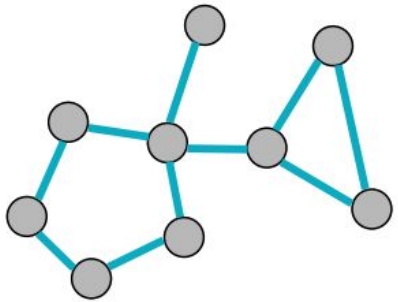
Properties: structure

Bitdefender

Structure - dependent
the processing should take into account the structure of the graphs

1. the processing should take into account how nodes are connected

CONNECTIVTY



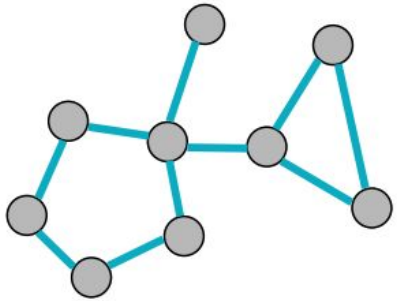
Properties: structure

Bitdefender

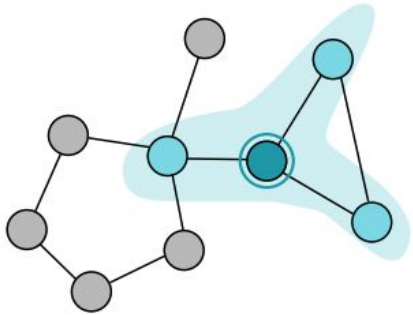
Structure - dependent
the processing should take into account the structure of the graphs

1. the processing should take into account how nodes are connected
2. a node should be influenced more by its neighbours

CONNECTIVTY



NEIGHBOURHOOD



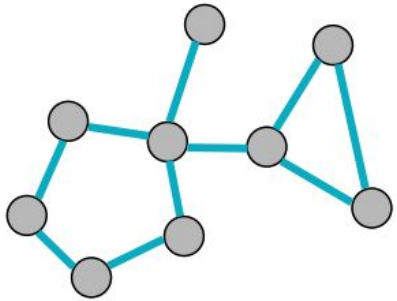
Properties: structure

Bitdefender

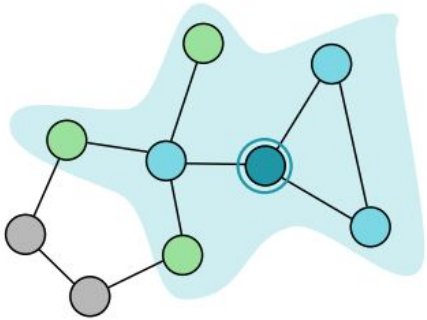
Structure - dependent
the processing should take into account the structure of the graphs

1. the processing should take into account how nodes are connected
2. a node should be influenced more by its neighbours

CONNECTIVTY



NEIGHBOURHOOD



Properties: permutation invariance and equivariance

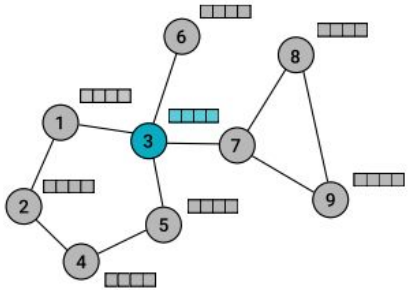
Bitdefender

There is no canonical order for the nodes of the graph.

Permutation invariance

The global output of the graph processing should be invariant to the order of the nodes.

$$f(PX, PAP') = f(X, A)$$



X

1				
2				
3				
4				
5				
6				
7				
8				
9				

A

	1	2	3	4	5	6	7	8	9
1	0	1	1	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0
3	1	0	0	0	1	1	1	0	0
4	0	1	0	0	1	0	0	0	0
5	0	0	1	1	0	0	0	0	0
6	0	0	1	0	0	0	0	0	0
7	0	0	1	0	0	0	0	1	1
8	0	0	0	0	0	0	1	0	1
9	0	0	0	0	0	0	1	1	0

Y

0.5	0.3	0.2

Properties: permutation invariance and equivariance

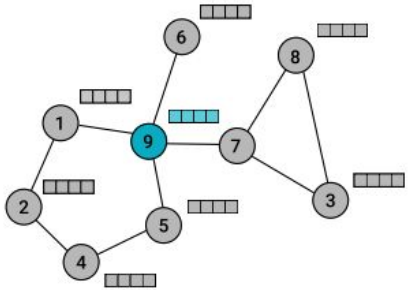
Bitdefender

There is no canonical order for the nodes of the graph.

Permutation invariance

The global output of the graph processing should be invariant to the order of the nodes.

$$f(PX, PAP') = f(X, A)$$



X

1				
2				
3				
4				
5				
6				
7				
8				
9				

A

	1	2	3	4	5	6	7	8	9
1	0	1	0	0	0	0	0	0	1
2	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	0	0
5	0	0	0	1	0	0	0	0	1
6	0	0	0	0	0	0	0	0	1
7	0	0	1	0	0	0	0	1	
8	0	0	1	0	0	0	1	0	0
9	1	0	0	0	1	1	1	0	0

Y

0.5	0.3	0.2

Properties: permutation invariance and equivariance

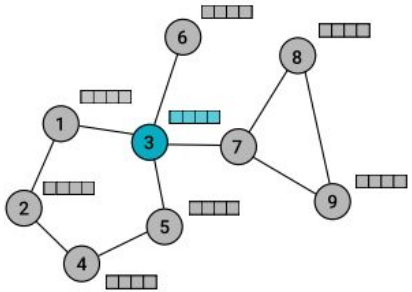
Bitdefender

There is no canonical order for the nodes of the graph.

Permutation equivariance

If we permute the input nodes of the graph, the nodes' output should be permuted in the same way.

$$f(PX, PAP') = Pf(X, A)$$



X

1			
2			
3			
4			
5			
6			
7			
8			
9			

A

	1	2	3	4	5	6	7	8	9
1	0	1	1	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0
3	1	0	0	0	1	1	1	0	0
4	0	1	0	0	1	0	0	0	0
5	0	0	1	1	0	0	0	0	0
6	0	0	1	0	0	0	0	0	0
7	0	0	1	0	0	0	0	1	1
8	0	0	0	0	0	0	1	0	1
9	0	0	0	0	0	0	1	1	0

Y

1			
2			
3			
4			
5			
6			
7			
8			
9			

Properties: permutation invariance and equivariance

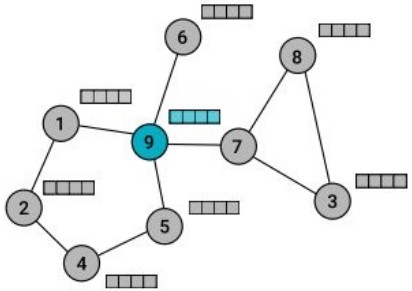
Bitdefender

There is no canonical order for the nodes of the graph.

Permutation equivariance

If we permute the input nodes of the graph, the nodes' output should be permuted in the same way.

$$f(PX, PAP') = Pf(X, A)$$



X

1				
2				
3				
4				
5				
6				
7				
8				
9				

A

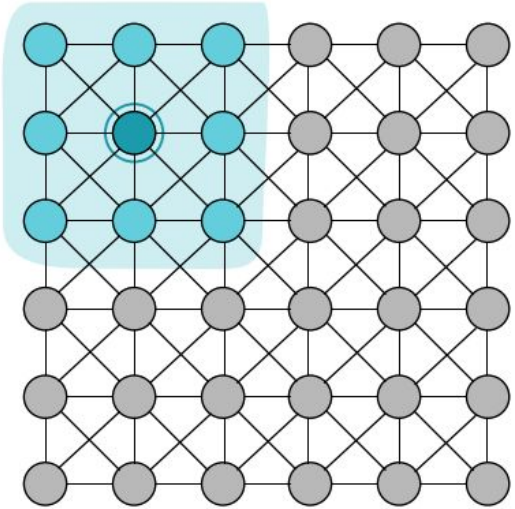
	1	2	3	4	5	6	7	8	9
1	0	1	0	0	0	0	0	0	1
2	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	0	0
5	0	0	0	1	0	0	0	0	1
6	0	0	0	0	0	0	0	0	1
7	0	0	1	0	0	0	0	1	1
8	0	0	1	0	0	0	1	0	0
9	1	0	0	0	1	1	1	0	0

Y

1			
2			
3			
4			
5			
6			
7			
8			
9			

Convolutional Network

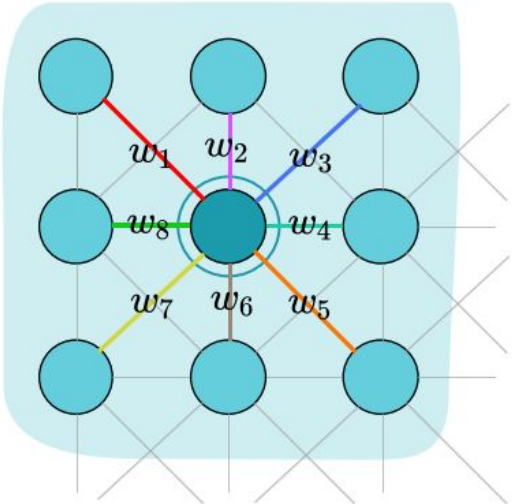
Bitdefender



- takes into account a **neighbourhood**
- the **structure is fixed**: a grid for 2D Conv or a sequence for 1D Conv
- the model is invariant to translations

Convolutional Network

Bitdefender



$$y_i = \sum_{j \in \mathcal{N}_i} w_j x_j$$

For a convolutional network the neighbourhood is

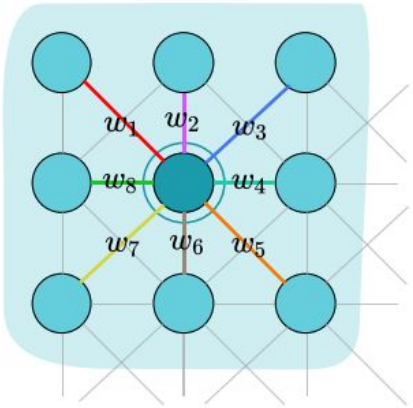
- **fixed**: for a $K \times K$ convolutional filter we combine exactly K^2 neighbours
- **ordered**: we can impose a canonical order among neighbours (left, right, up, down)

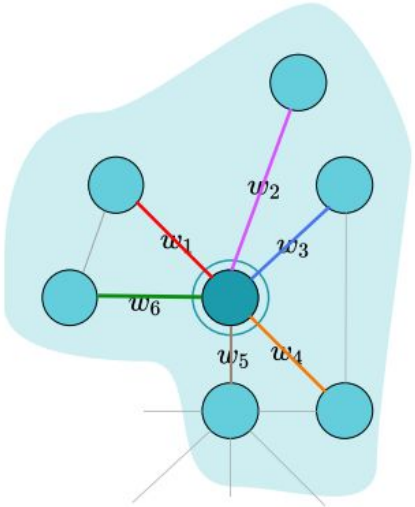
Convolutional Network

Bitdefender

$$y_i = \sum_{j \in \mathcal{N}_i} w_j x_j$$

Can we do the same for graphs?



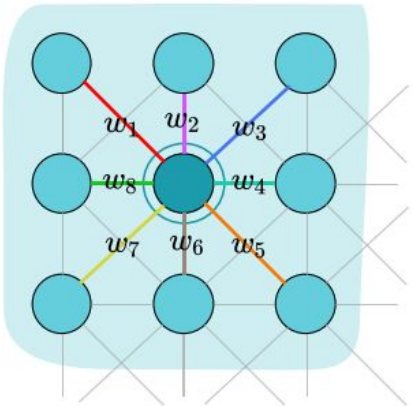


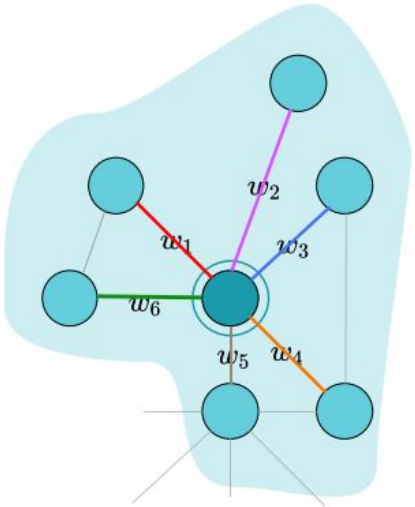
Convolutional Network

Bitdefender

$$y_i = \sum_{j \in \mathcal{N}_i} w_j x_j$$

$$y_i = \sum_{j \in \mathcal{N}_i} w_j \, x_j$$





- can’t have variable number of weights
- have to establish an order

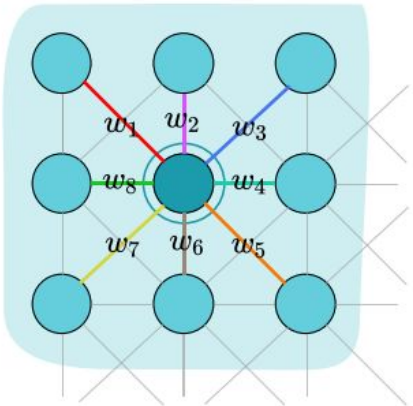
Convolutional Network

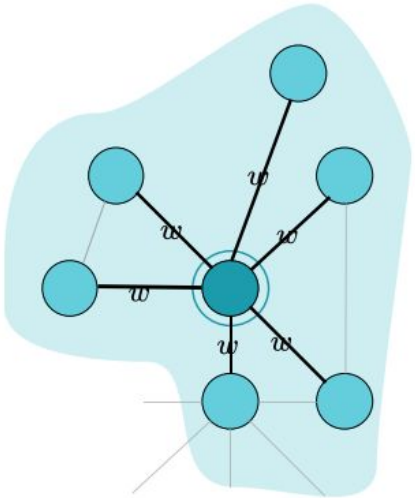
Bitdefender

$$y_i = \sum_{j \in \mathcal{N}_i} w_j x_j$$

$$y_i = \sum_{j \in \mathcal{N}_i} w\, x_j$$

- Solution: same w for all nodes

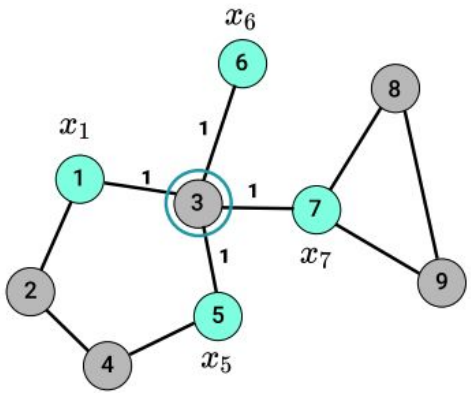




Graph Propagation

Bitdefender

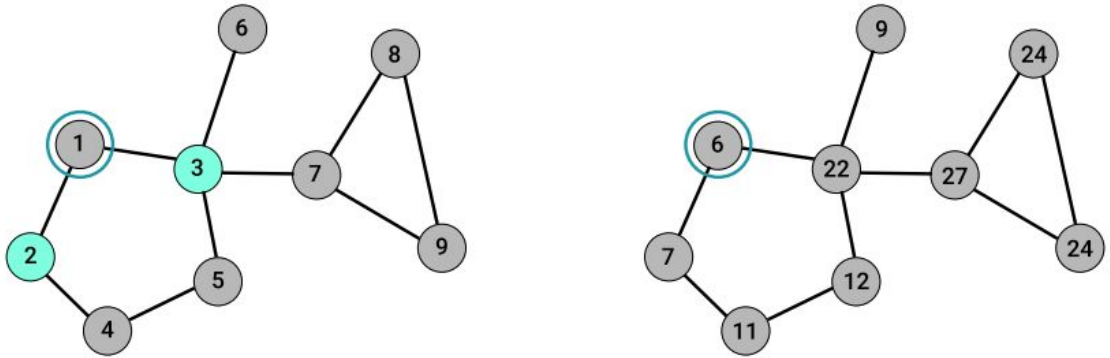
Simple graph representation (set $w = 1$): $y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j$



Graph Propagation

Bitdefender

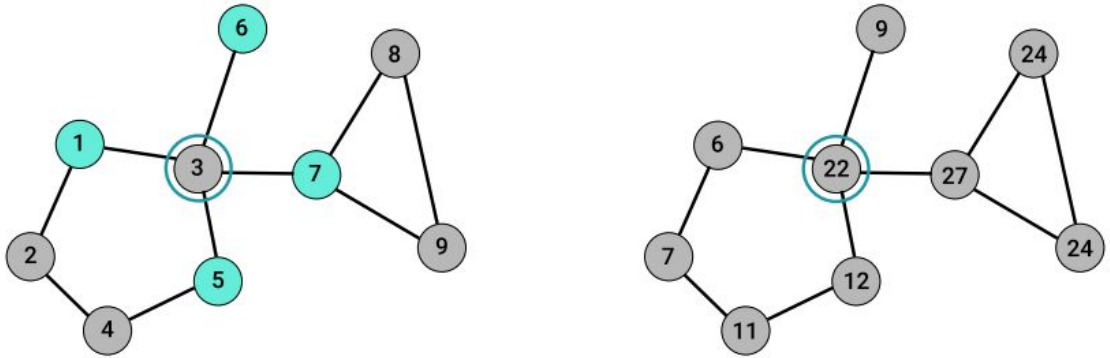
Simple graph representation (set $w = 1$): $y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j$



Graph Propagation

Bitdefender

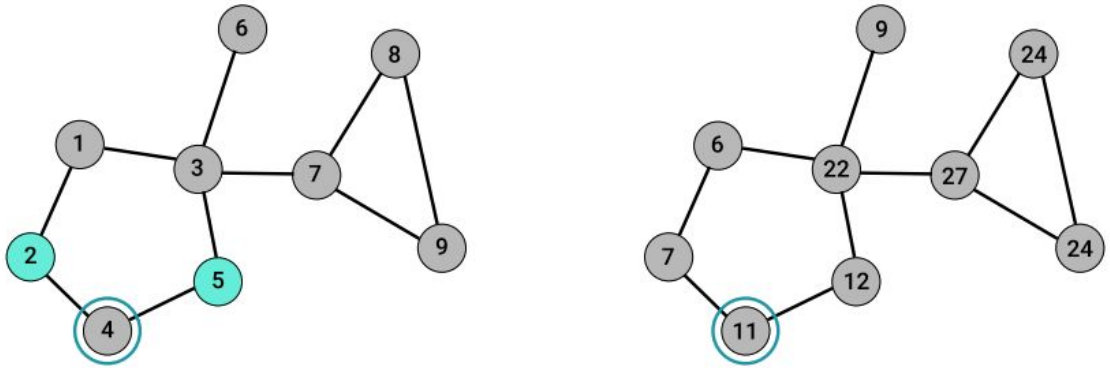
Simple graph representation (set $w = 1$): $y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j$



Graph Propagation

Bitdefender

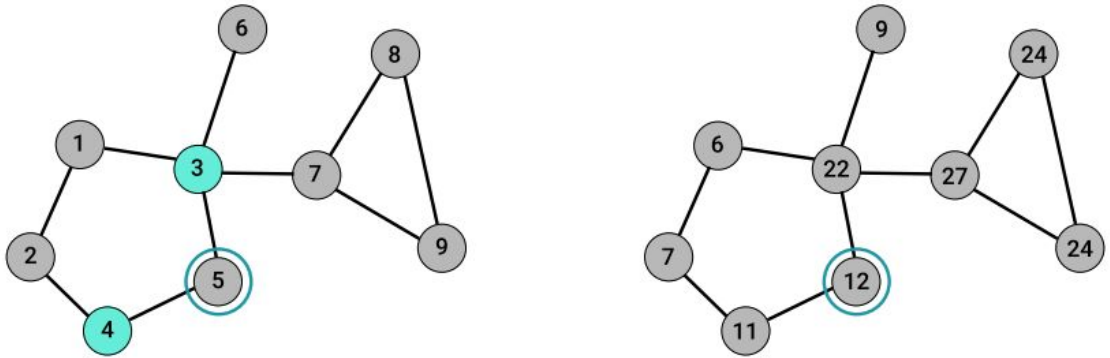
Simple graph representation (set $w = 1$): $y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j$



Graph Propagation

Bitdefender

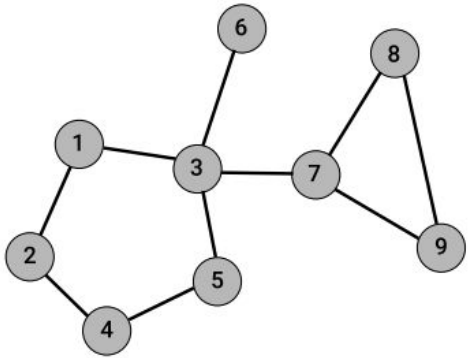
Simple graph representation (set $w = 1$): $y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j$

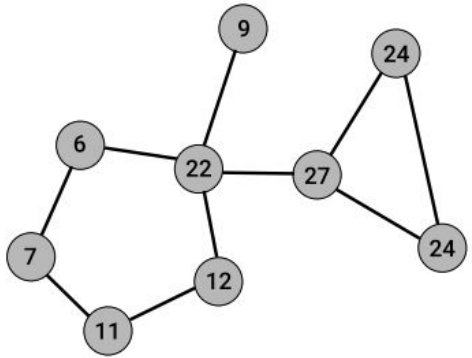


Graph Propagation

Bitdefender

Simple graph representation (set $w = 1$): $y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j$

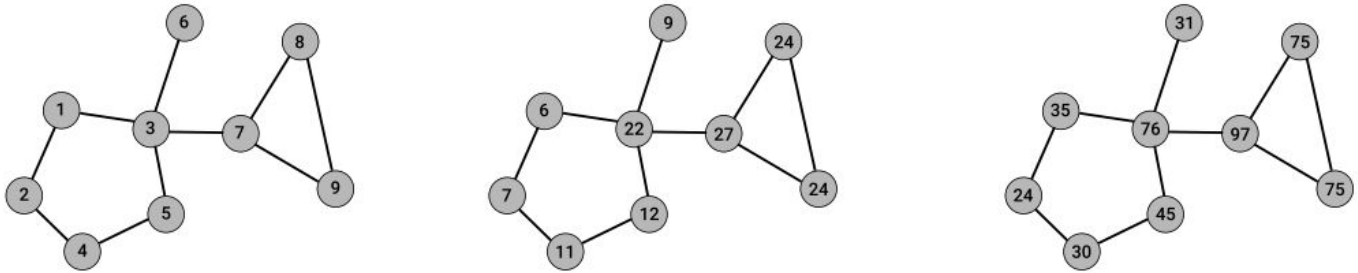




Graph Propagation

Bitdefender

Simple graph propagation (set $w = 1$): $y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j$

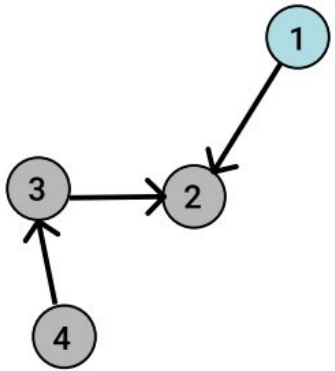


- if applied iteratively, it takes into account the structure

Simplest Graph Propagation

Bitdefender

$y_i = \sum_{j \in \mathcal{N}_i} x_j$ can be rewritten in a compact, matrix form as $Y = AX$



$A \in \mathbb{R}^{N \times N}$ $X \in \mathbb{R}^{N}$ $Y \in \mathbb{R}^{N}$

0	0	0	0
1	0	1	0
0	0	0	1
0	0	0	0

1
2
3
4

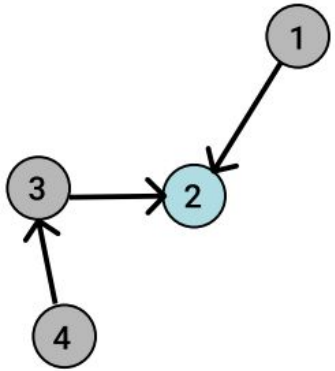
$=$

0

Simplest Graph Propagation

Bitdefender

$y_i = \sum_{j \in \mathcal{N}_i} x_j$ can be rewritten in a compact, matrix form as $Y = AX$



$A \in \mathbb{R}^{N \times N}$ $X \in \mathbb{R}^{N}$ $Y \in \mathbb{R}^{N}$

0	0	0	0
1	0	1	0
0	0	0	1
0	0	0	0

1
2
3
4

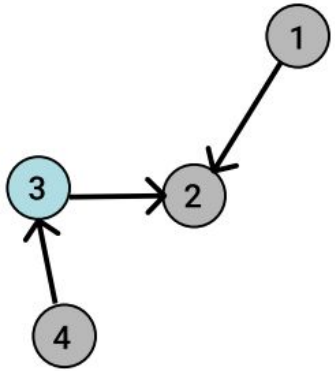
$=$

0
1+3

Simplest Graph Propagation

Bitdefender

$y_i = \sum_{j \in \mathcal{N}_i} x_j$ can be rewritten in a compact, matrix form as $Y = AX$



$A \in \mathbb{R}^{N \times N}$ $X \in \mathbb{R}^{N}$ $Y \in \mathbb{R}^{N}$

0	0	0	0
1	0	1	0
0	0	0	1
0	0	0	0

1
2
3
4

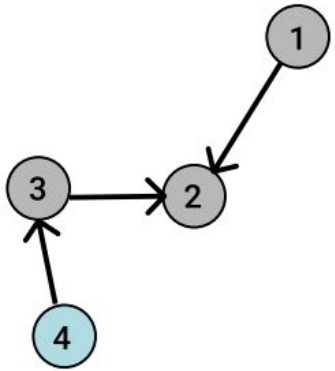
$=$

0
1+3
4

Simplest Graph Propagation

Bitdefender

$y_i = \sum_{j \in \mathcal{N}_i} x_j$ can be rewritten in a compact, matrix form as $Y = AX$



$A \in \mathbb{R}^{N \times N}$ $X \in \mathbb{R}^{N}$ $Y \in \mathbb{R}^{N}$

0	0	0	0
1	0	1	0
0	0	0	1
0	0	0	0

1
2
3
4

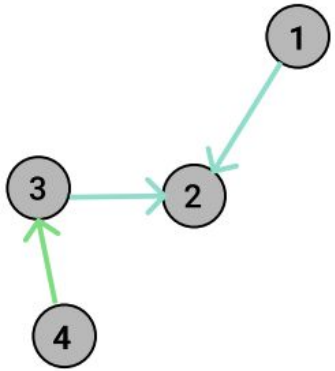
$=$

0
1+3
4
0

Simplest Graph Propagation

Bitdefender

$y_i = \sum_{j \in \mathcal{N}_i} x_j$ Nodes could have high-dimensional representation $X \in R^{N \times D}$



$A \in \mathbb{R}^{N \times N}$ $X \in \mathbb{R}^{N \times D}$ $Y \in \mathbb{R}^{N \times D}$

0	0	0	0
1	0	1	0
0	0	0	1
0	0	0	0

x_1
x_2
x_3
x_4

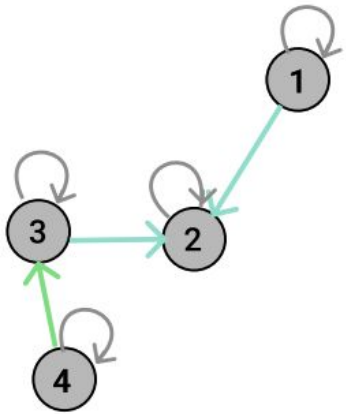
$=$

0
x_1+x_3
x_4
0

Simplest Graph Propagation

Bitdefender

$y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j$ We should take into account also the current node - self-loops.



$A \in \mathbb{R}^{N \times N}$ $X \in \mathbb{R}^{N \times D}$ $Y \in \mathbb{R}^{N \times D}$

1	0	0	0
1	1	1	0
0	0	1	1
0	0	0	1

x_1
x_2
x_3
x_4

$=$

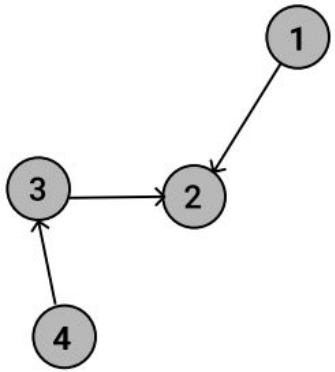
x_1
$x_1+x_2+x_3$
x_3+x_4
x_4

Simplest Graph Propagation

Bitdefender

To combine more complex representations:

$y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j \quad \rightarrow \quad y_i = x_i W + \sum_{j \in \mathcal{N}_i} x_j W$



$X \in \mathbb{R}^{N \times D} \quad W \in \mathbb{R}^{D \times C} \quad Y \in \mathbb{R}^{N \times C}$

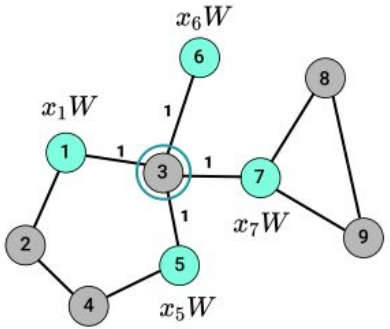
$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \begin{bmatrix} \quad \end{bmatrix} = \begin{bmatrix} x_1 W \\ x_2 W \\ x_3 W \\ x_4 W \end{bmatrix}$$

Simplest Graph Propagation

Bitdefender

To combine more complex representations:

$y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j \quad \rightarrow \quad y_i = x_i W + \sum_{j \in \mathcal{N}_i} x_j W$



The operations performed in the graph could be rewritten as:

$$Y = AXW$$

Iteratively, for more layers:

$$Y = A\sigma(AXW_1)W_2)$$

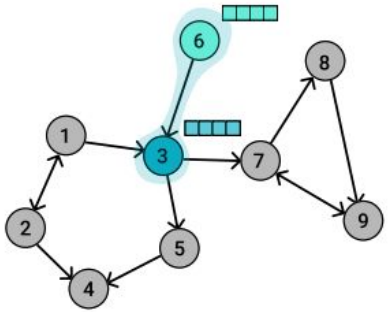
$$Y = A\sigma...A\sigma(AXW_1)W_2)..W_n$$

GNNs: Message Passing Framework - Send

Bitdefender

Send Function

- for each pair of 2 connected nodes, create a **message**



$$m_{ij} = f_{msg}(x_i, x_j) \in \mathbb{R}^C \quad \forall (i,j) \in \mathcal{E}$$

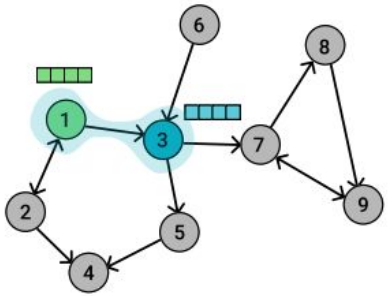
$m_{3,6} = f_{msg}($, $)$

GNNs: Message Passing Framework - Send

Bitdefender

Send Function

- for each pair of 2 connected nodes, create a **message**



$$m_{ij} = f_{msg}(x_i, x_j) \in \mathbb{R}^C \quad \forall (i,j) \in \mathcal{E}$$

$m_{3,6} = f_{msg}($, $)$

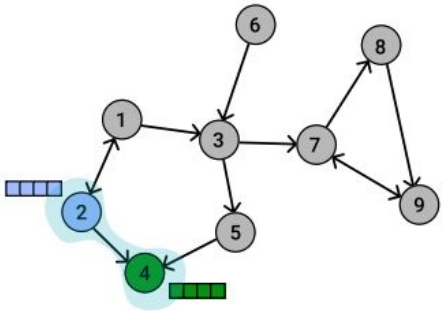
$m_{3,1} = f_{msg}($, $)$

GNNs: Message Passing Framework - Send

Bitdefender

Send Function

- for each pair of 2 connected nodes, create a **message**



$$m_{ij} = f_{msg}(x_i, x_j) \in \mathbb{R}^C \quad \forall (i,j) \in \mathcal{E}$$

$m_{3,6} = f_{msg}($, $)$

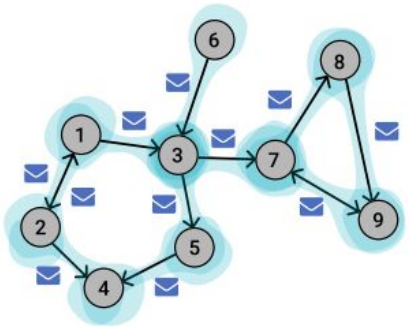
$m_{3,1} = f_{msg}($, $)$

$m_{4,2} = f_{msg}($, $)$

GNNs: Message Passing Framework - Send

Bitdefender

- f_{msg} is a learnable function (e.g. an MLP)
- its parameters are shared between each pair of nodes



Learnable funcion

$$m_{ij} = \overbrace{f_{msg}(x_i, x_j)} \in \mathbb{R}^C \quad \forall (i,j) \in \mathcal{E}$$

$$m_{3,6} = f_{msg}(\,,\,)$$
$$m_{3,1} = f_{msg}(\,,\,)$$
$$\dots$$
$$m_{4,2} = f_{msg}(\,,\,)$$

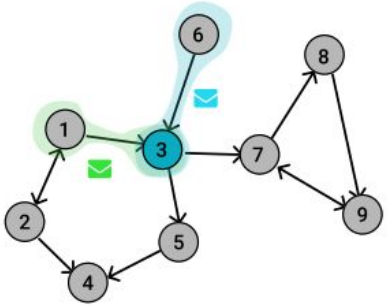
Same parameters

GNNs: Message Passing Framework - Aggregation

Bitdefender

Aggregation Function

For each node i, **aggregate** the incoming messages from all its neighbours.



$$h_i = f_{agg}(\{m_{ij} | \forall j \in \mathcal{N}_i\})$$

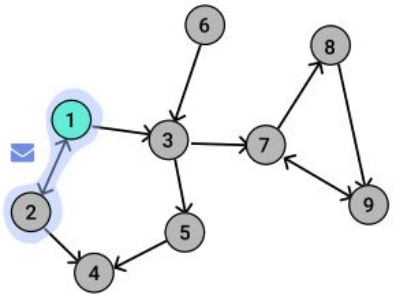
$$h_3 = f_{agg}(\{ \text{✉}, \text{✉} \})$$

GNNs: Message Passing Framework - Aggregation

Bitdefender

Aggregation Function

For each node i, **aggregate** the incoming messages from all its neighbours.



$$h_i = f_{agg}(\{m_{ij} | \forall j \in \mathcal{N}_i\})$$

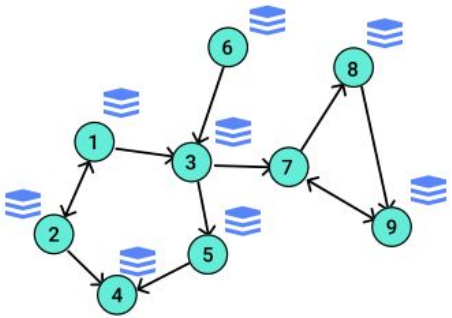
$h_3 = f_{agg}(\{$ ✉ , ✉ $\})$

$h_1 = f_{agg}(\{$ ✉ $\})$

GNNs: Message Passing Framework - Aggregation

Bitdefender

- aggregate incoming messages with the function f_{agg}: eg. sum, mean, max, min
- it should be **invariant to the order** of the nodes and should **allow a variable number** of messages



operator

$$h_i = \overbrace{f_{agg}}\ (\{m_{ij} | \forall j \in \mathcal{N}_i\}) \in \mathbb{R}^C$$

$$h_3 = f_{agg}(\{\blacksquare, \blacksquare\})$$

$$\dots$$

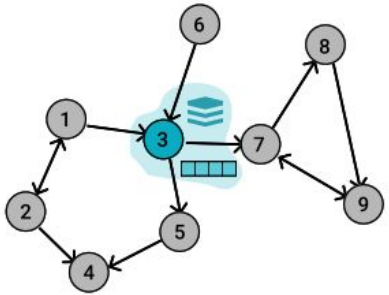
$$h_1 = f_{agg}(\{\blacksquare\})$$

GNNs: Message Passing Framework - Update

Bitdefender

Update Function

For each node i, **update** its representation using the aggregated message.



$$\tilde{x}_i = f_{upd}(x_i, h_i)$$

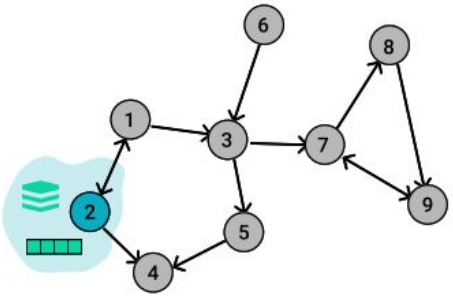
$\tilde{x}_3 =$ $f_{upd}($  ,  $)$

GNNs: Message Passing Framework - Update

Bitdefender

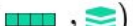
Update Function

For each node i, **update** its representation using the aggregated message.



$$\tilde{x}_i = f_{upd}(x_i, h_i)$$

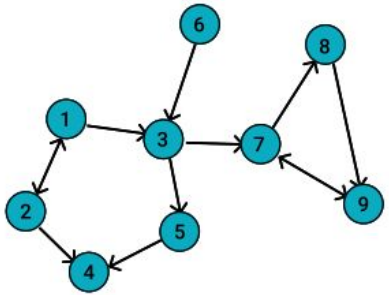
$\tilde{x}_3 = f_{upd}($  $)$

$\tilde{x}_2 = f_{upd}($  $)$

GNNs: Message Passing Framework - Update

Bitdefender

- f_{upd} is a learnable function (e.g. an MLP)
- its parameters are shared between all the nodes



Learnable function

$$\tilde{x}_i = \overbrace{f_{upd}(x_i, h_i)} \in \mathbb{R}^C$$

$\tilde{x}_3 = f_{upd}($  $)$

$\cdots$

$\tilde{x}_2 = f_{upd}($  $)$

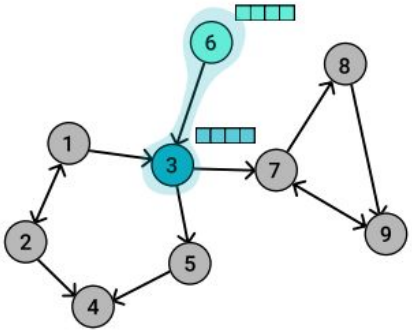
Same parameters

GNNs - Overview

Bitdefender

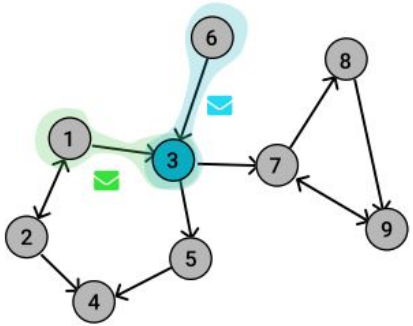
1. Send

$$m_{ij} = f_{msg}(x_i, x_j)$$



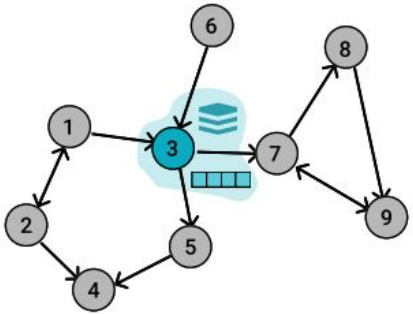
2. Aggregate

$$H_i = f_{agg}(\{m_{ij} | \forall j \in \mathcal{N}_i\})$$



3. Update

$$\tilde{x}_i = f_{upd}(x_i, H_i)$$



General GNN framework

Bitdefender

$$f_{upd}\{x_i,\ f_{agg}\{\ f_{msg}(x_i, x_j)\ |\forall j \in \mathcal{N}_i\}\ \}$$

Depending on how the 3 functions are instantiated, different architectures could be obtained:

Convolutional GNNs

$$f_{upd}(x_i, \bigoplus_{\forall j \in \mathcal{N}_i} \{\phi(x_j)\})$$

Attention GNNs

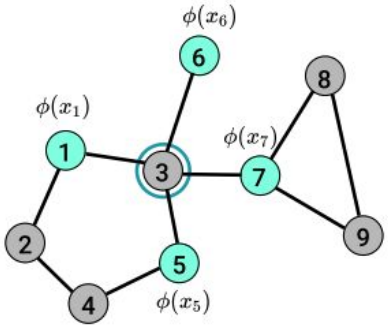
$$f_{upd}(x_i, \bigoplus_{\forall j \in \mathcal{N}_i} \{\alpha(x_i, x_j)\phi(x_j))$$

Message Passing

$$f_{upd}(x_i, \bigoplus_{\forall j \in \mathcal{N}_i} \{\phi(x_i, x_j)\})$$

Graph Convolutional Network

Bitdefender

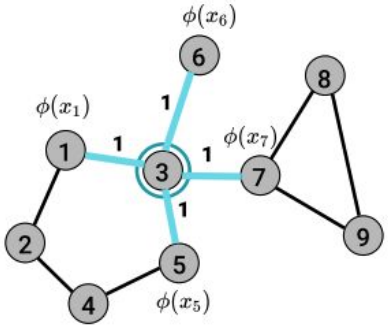


$$y_i = f_{upd}(x_i, \bigoplus_{\forall j \in \mathcal{N}_i} \{ \phi(x_j) \})$$

- messages depend only on the source nodes

Graph Convolutional Network

Bitdefender



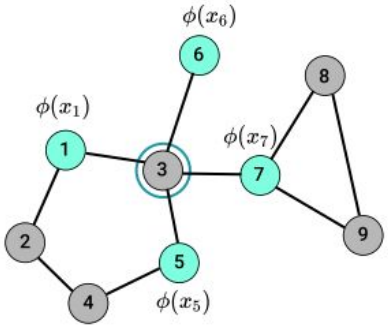
$$y_i = f_{upd}(x_i, \bigoplus_{\forall j \in \mathcal{N}_i} \{\phi(x_j)\})$$

- messages depend only on the source nodes
- aggregation function is implemented as a sum/mean operation
- aggregation could be normalized according to the nodes' degree: $\frac{1}{\sqrt{deg(i)deg(j)}}$

Matrix form: $Y = \sigma(\tilde{A}XW)$

Graph Attention Network

Bitdefender

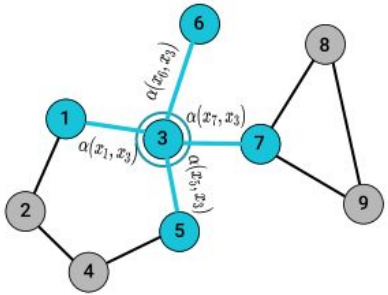


$$y_i = f_{upd}(x_i, \bigoplus_{\forall j \in \mathcal{N}_i} \{\alpha(x_i, x_j)\{\ \phi(x_j)\ \})$$

- messages depend only on the source nodes

Graph Attention Network

Bitdefender



$$y_i = f_{upd}(x_i, \{ \bigoplus_{\forall j \in \mathcal{N}_i} \{\alpha(x_i, x_j)\, \phi(x_j)\})$$

- messages depend only on the source nodes
- aggregation function is based on attention mechanism

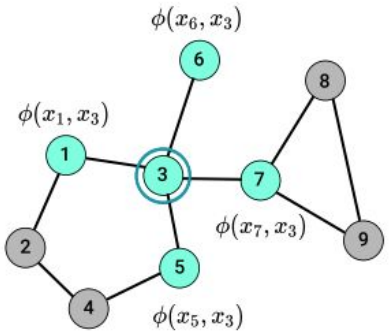
GAT: $\alpha(x_i, x_j) \propto \text{ReLU}(a[x_i W_1, x_j W_2]^T) \in \mathbb{R}$

Self-Attention: $\alpha(x_i, x_j) \propto x_i W_1 (x_j W_2)^T \in \mathbb{R}$

- the model is able to learn the desired structure

Message Passing Neural Network

Bitdefender

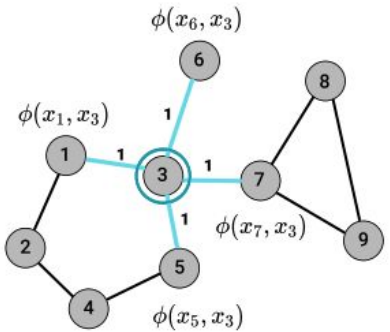


$$y_i = f_{upd}(x_i, \bigoplus_{\forall j \in \mathcal{N}_i} \{ \phi(x_i, x_j) \})$$

- messages depend on both source and destination
- if edge features are available, the message could also take them into account

Message Passing Neural Network

Bitdefender



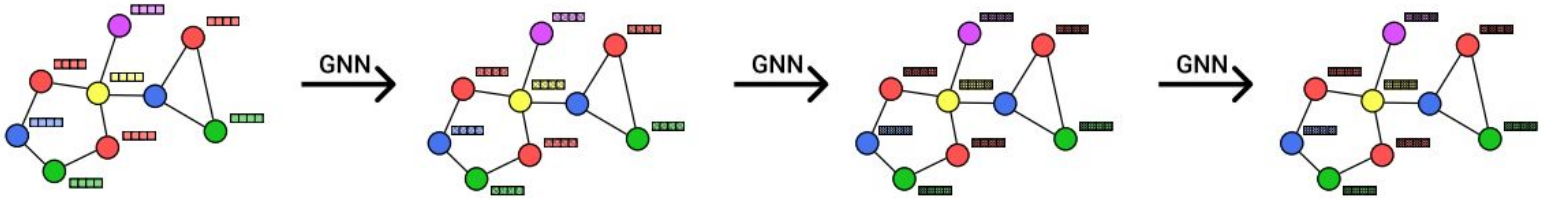
$$y_i = f_{upd}(x_i, \bigoplus_{\forall j \in \mathcal{N}_i} \{\phi(x_i, x_j)\})$$

- messages depend on both source and destination
- if edge features are available, the message could also take them into account
- aggregation function is implemented as a sum/mean operation

Multiple Layers

Bitdefender

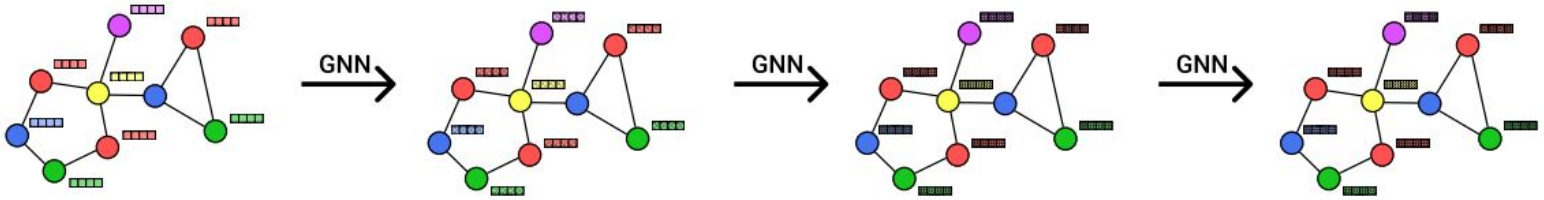
- for a more powerful representation, we can stack multiple layers



Multiple Layers

Bitdefender

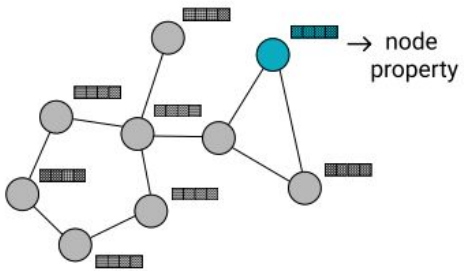
- for a more powerful representation, we can stack multiple layers
- each layer increases the receptive field of each node



RECEPTIVE FIELD:

Graph Output - Node Level

Bitdefender



- predict an output y_i from each node

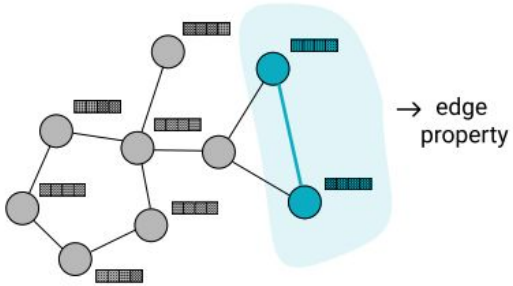
$$y_i = f_{output}(\tilde{x_i}) \quad \in \mathbb{R}^K$$

- the loss function is applied for each node in the graph

$$\mathcal{L} = \sum_{i \in \mathcal{V}} \mathcal{L}_i(y_i, l_i)$$

Graph Output - Edge Level

Bitdefender



- predict an output y_{ij} from each pair of nodes

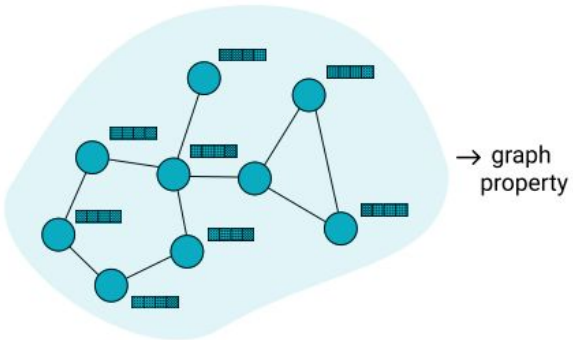
$$y_{ij} = f_{output}(\tilde{x_i}, \tilde{x_j}) \quad \in \mathbb{R}^K$$

- the loss function is applied for each edge in the graph

$$\mathcal{L} = \sum_{(i,j)\in\mathcal{E}} \mathcal{L}_i(y_{ij}, l_{ij})$$

Graph Output - Graph Level

Bitdefender



- predict a single output y for the whole graph

$$y = f_{readout}(\{\tilde{x}_i | \forall i \in \mathcal{V}\}) \quad \in \mathbb{R}^K$$

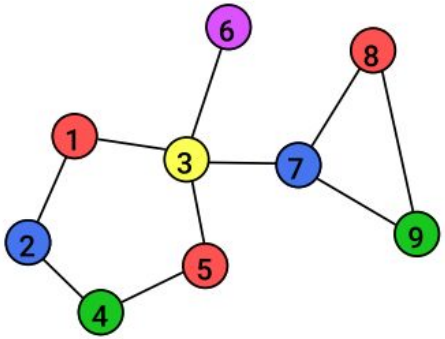
- $f_{readout}$ could be a simple order-invariant aggregator (e.g. sum, mean), or more complex graph pooling mechanisms
- the loss function is applied for each graph in the dataset

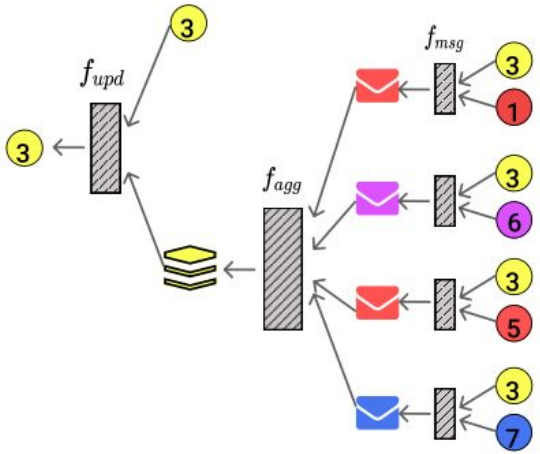
$$\mathcal{L} = \mathcal{L}_i(y, l)$$

Learning

Bitdefender

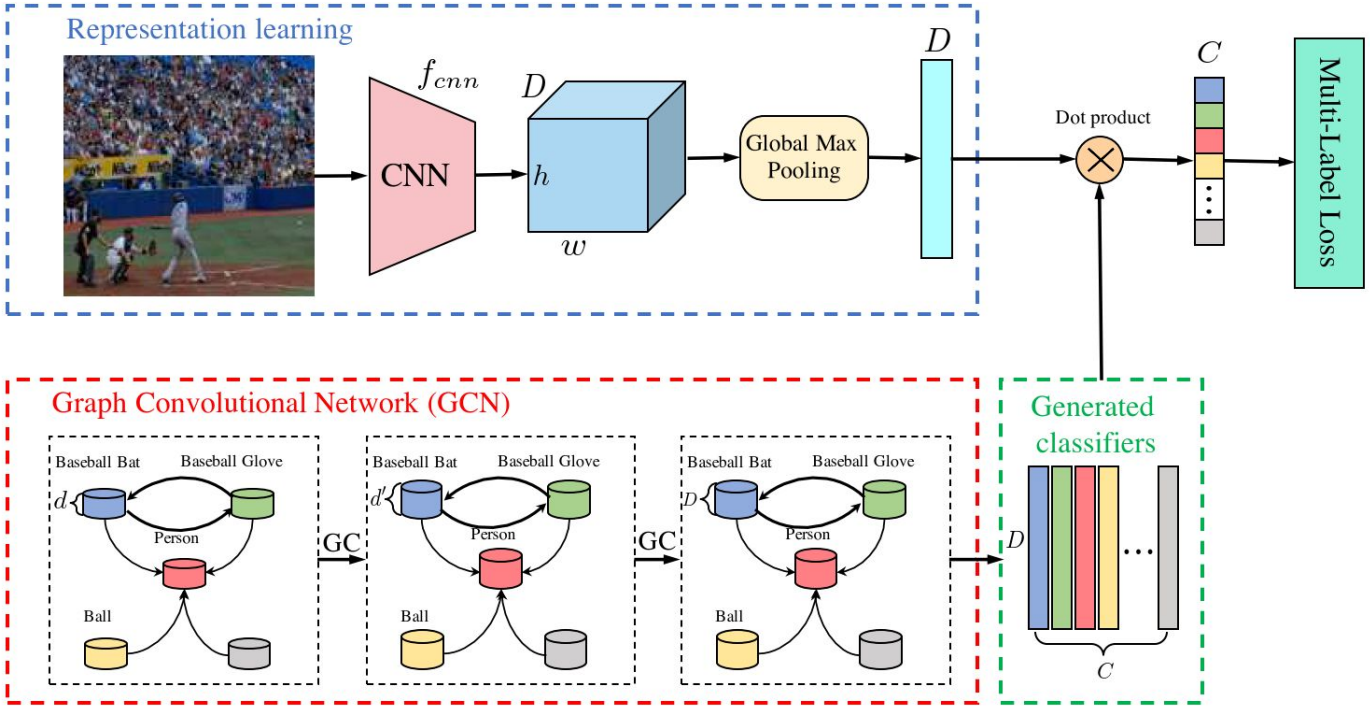
- the output of a GNN for a node i is obtained by applying a **sequence of operations** on the initial nodes
- all the operations along the sequence should be **differentiable**





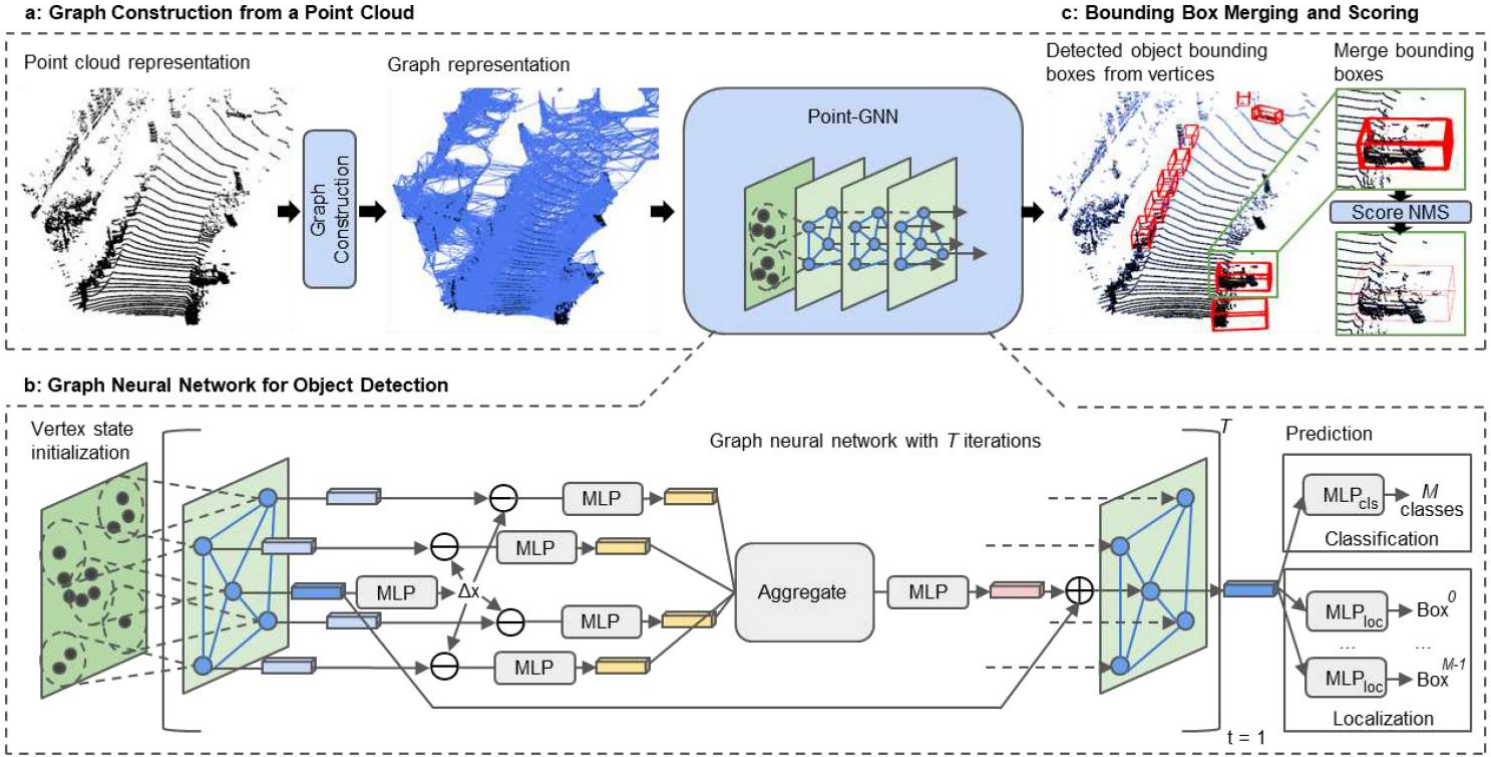
GNN applications in vision

Multi-label image classification



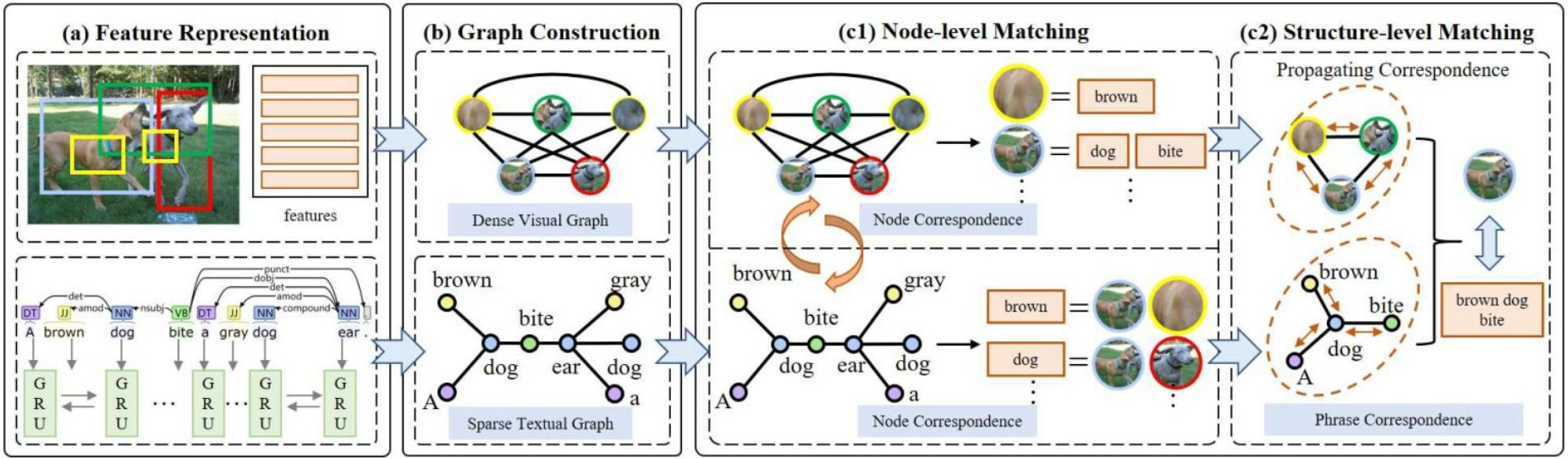
Chen, Zhao-Min, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. "Multi-label image recognition with graph convolutional networks." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 5177-5186. 2019.

Object detection from 3D point clouds



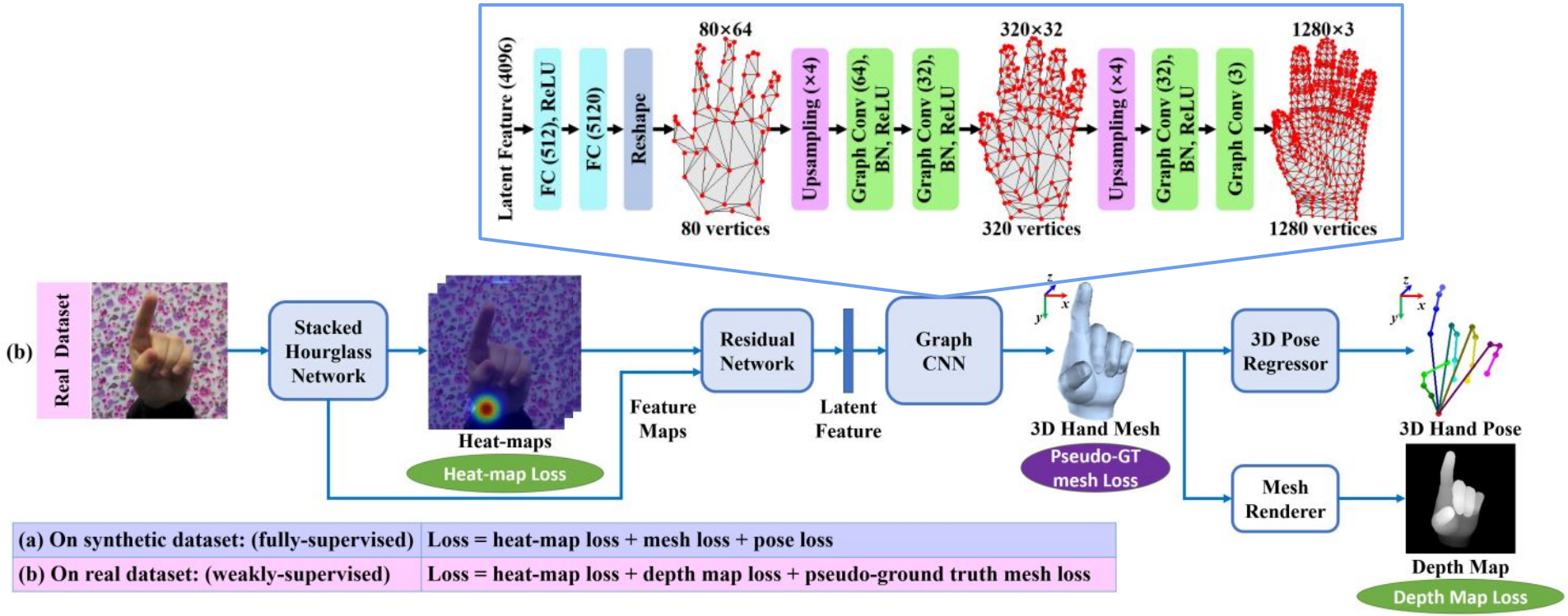
Shi, Weijing, and Raj Rajkumar. "Point-gnn: Graph neural network for 3d object detection in a point cloud." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 1711-1719. 2020.

Image-text matching



Liu, Chunxiao, Zhendong Mao, Tianzhu Zhang, Hongtao Xie, Bin Wang, and Yongdong Zhang. "Graph structured network for image-text matching." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10921-10930. 2020.

3D hand mesh reconstruction



(a) On synthetic dataset: (fully-supervised)	Loss = heat-map loss + mesh loss + pose loss
(b) On real dataset: (weakly-supervised)	Loss = heat-map loss + depth map loss + pseudo-ground truth mesh loss

Ge, Liuhao, Zhou Ren, Yuncheng Li, Zehao Xue, Yingying Wang, Jianfei Cai, and Junsong Yuan. "3d hand shape and pose estimation from a single rgb image." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10833-10842. 2019.