Graph Neural Networks

Introduction

Iulia Duta

Andrei Nicolicioiu

UNIVERSITATEA DIN BUCUREȘTI
VIRTUTE ET SAPIENTIA

Bitdefender

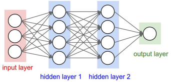
May 2021

Choose your model

Bitdefender

UNSTRUCTURED

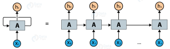
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	Iris setosa
4.9	3.0	1.4	0.2	Iris setosa
7.0	3.2	4.7	1.4	Iris versicolor
6.4	3.2	4.5	1.5	Iris versicolor
6.3	3.3	6.0	2.5	Iris virginica
5.8	3.3	6.0	2.5	Iris virginica



MLP

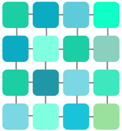
SEQUENTIAL

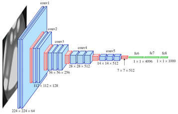
Have a nice day! :)



RNN

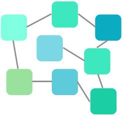
GRID

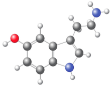


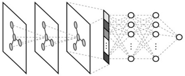


CNN

RELATIONAL STRUCTURE





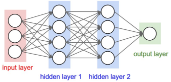


GNN

Choose your model

UNSTRUCTURED

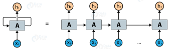
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	Iris setosa
4.9	3.0	1.4	0.2	Iris setosa
7.0	3.2	4.7	1.4	Iris versicolor
6.4	3.2	4.5	1.5	Iris versicolor
6.3	3.3	6.0	2.5	Iris virginica
5.8	3.3	6.0	2.5	Iris virginica



MLP

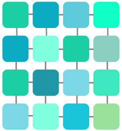
SEQUENTIAL

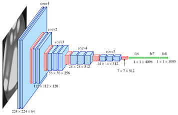
Have a nice day! :)



RNN

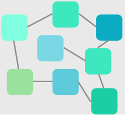
GRID

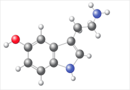


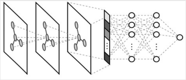


CNN

RELATIONAL STRUCTURE

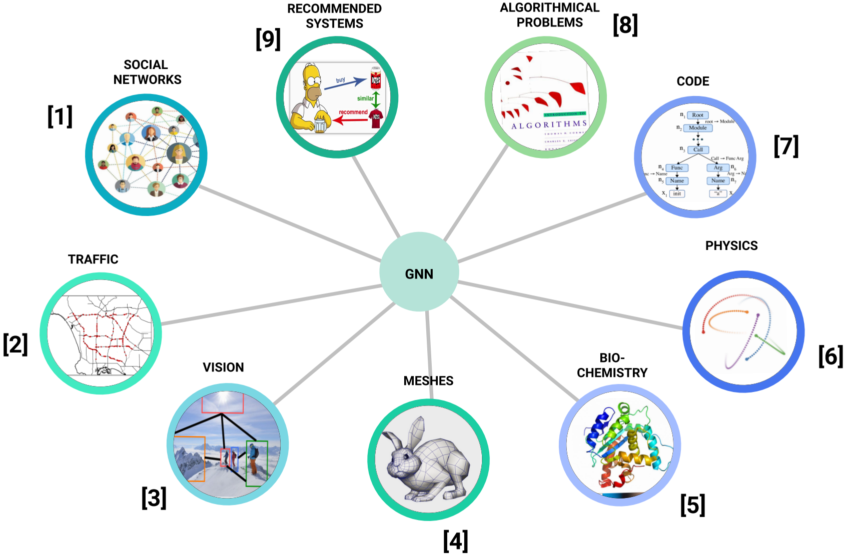






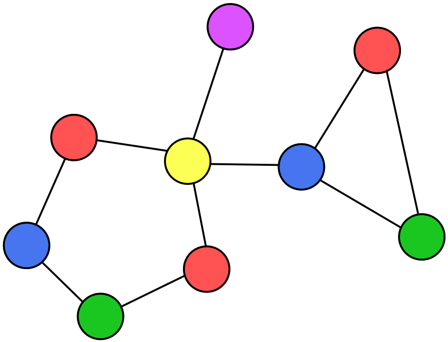
GNN

Tasks


GNN
SOCIAL NETWORKS
[1]
TRAFFIC
[2]
VISION
[3]
MESHES
[4]
BIO-CHEMISTRY
[5]
PHYSICS
[6]
CODE
[7]
ALGORITHMICAL PROBLEMS
[8]
RECOMMENDED SYSTEMS
[9]
buy
similar
recommend
ALGORITHMS

Data: Graph Structure

Bitdefender



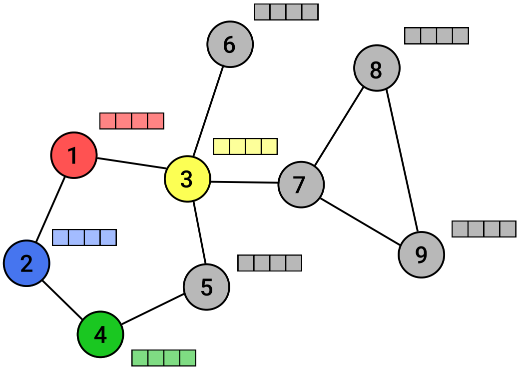
Tasks where we have access or we can create a graph structure.

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.

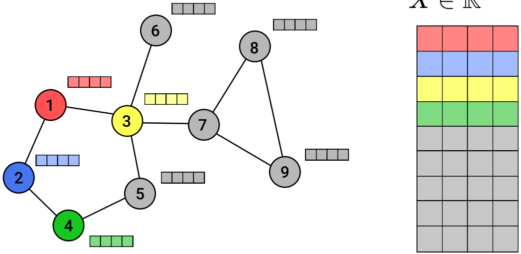
A graph G is characterized by:

- 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 $x_i \in \mathbb{R}^D$

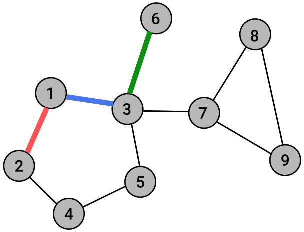
Data: Graph Structure - Nodes

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



- 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

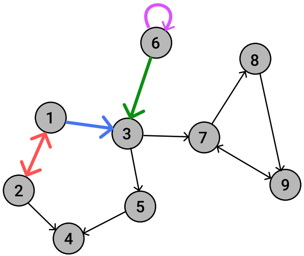


$$A \in \mathbb{R}^{N \times N}$$

	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



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

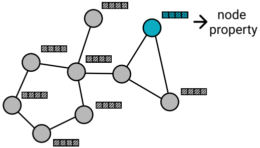
	1	2	3	4	5	6	7	8	9
1	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0
3	1	0	0	0	0	1	0	0	0
4	0	1	0	0	1	0	0	0	0
5	0	0	1	1	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	0	1
8	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	1	1	0

- 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

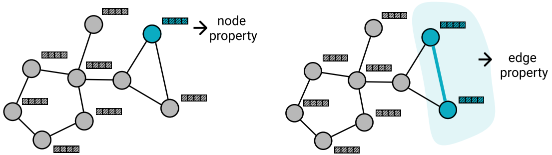
- Based on the node features (X) and the graph structure (A), we want to learn a representation of the graph.
- Depending on the task, the representation could be:
  1. node level: $Y \in \mathbb{R}^{N \times K}$



GNNs Goal

Bitdefender

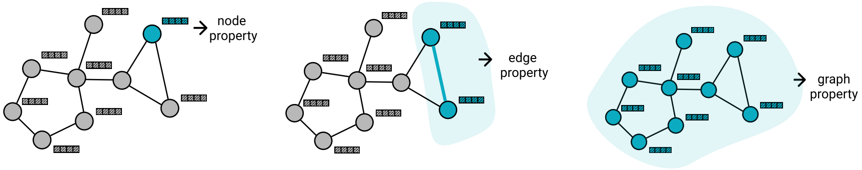
- Based on the node features (X) and the graph structure (A), we want to learn a representation of the graph.
- Depending on the task, the representation could be:
    1. node level: $Y \in \mathbb{R}^{N \times K}$
    2. edge level: $Y \in \mathbb{R}^{M \times K}$



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: $Y \in \mathbb{R}^{K}$

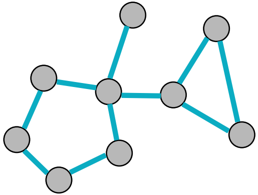


Properties: structure

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



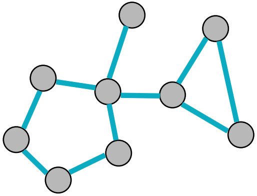
Properties: structure

Bitdefender

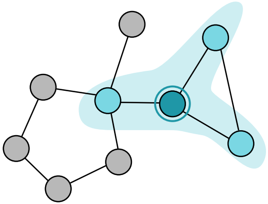
Structure - dependent
the processing should take into account the structure of the graphs

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

CONNECTIVTY



NEIGHBOURHOOD

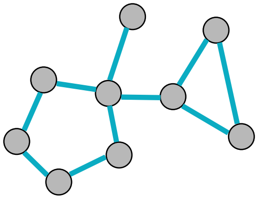


Properties: structure

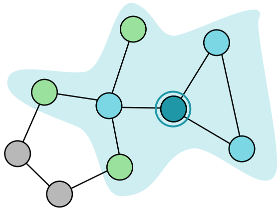
Structure - dependent
the processing should take into account the structure of the graphs

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

CONNECTIVTY



NEIGHBOURHOOD



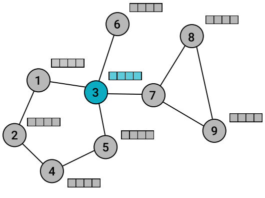
Properties: permutation invariance and equivariance

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

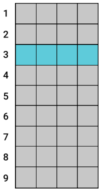
Permutation invariance

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

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



X



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

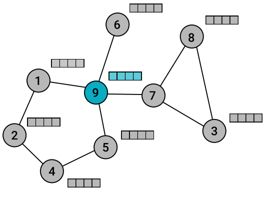
Bitdefender

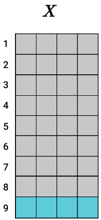
There is no canonical order for the nodes of the graph.

Permutation invariance

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

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





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

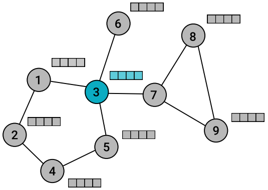
Properties: permutation invariance and equivariance

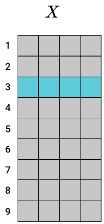
There is no canonical order for the nodes of the graph.

Permutation equivariance

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

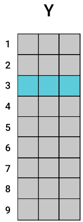
$$f(PX, PAP') = Pf(X, A)$$





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



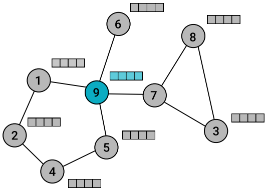
Properties: permutation invariance and equivariance

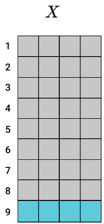
There is no canonical order for the nodes of the graph.

Permutation equivariance

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

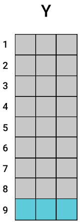
$$f(PX, PAP') = Pf(X, A)$$



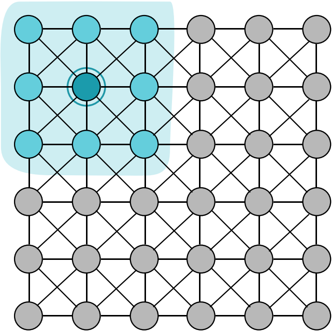


A

	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



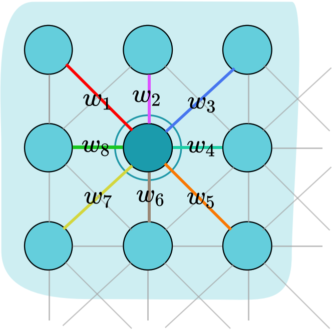
Convolutional Network



- 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



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

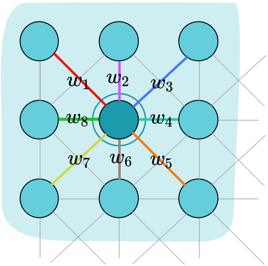
For a convolutional network the neighbourhood is

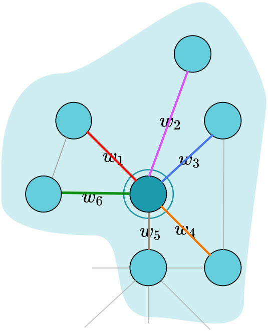
- **fixed**: for a $K \times K$ convolutional filter we combine exactly K^2 neighbours
- **ordered**: we can impose a canonical order among neighbours (left, right, up, down)

Convolutional Network

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

Can we do the same for graphs?



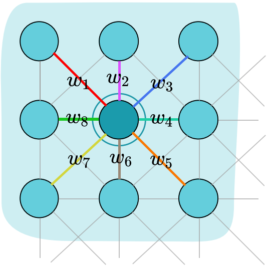


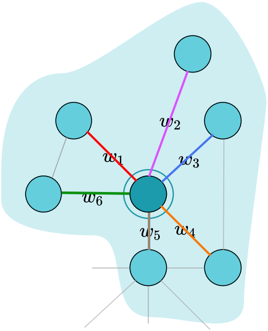
Convolutional Network

Bitdefender

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

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





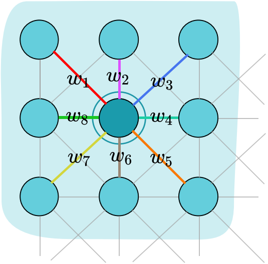
- can't have variable number of weights
- have to establish an order

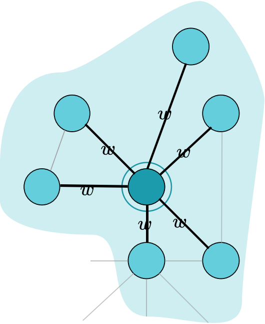
Convolutional Network

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

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

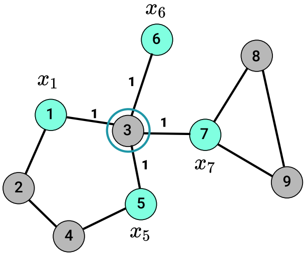
- Solution: same w for all nodes





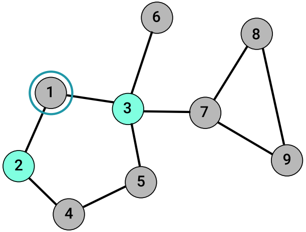
Graph Propagation

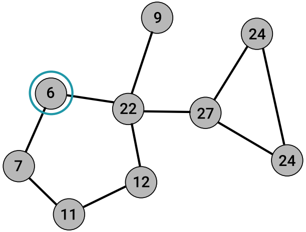
Simple graph representation (set $w = 1$): $$y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j$$



Graph Propagation

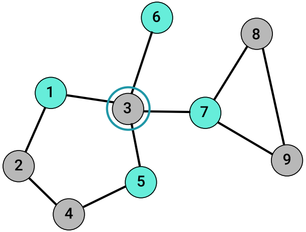
Simple graph representation (set $w = 1$): $y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j$

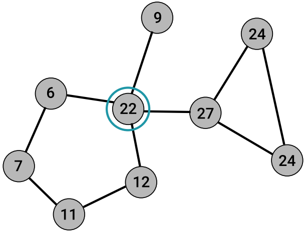




Graph Propagation

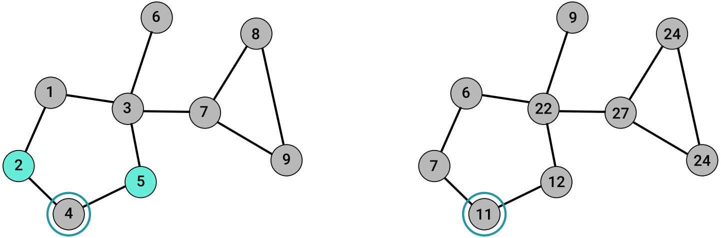
Simple graph representation (set $w = 1$): $y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j$





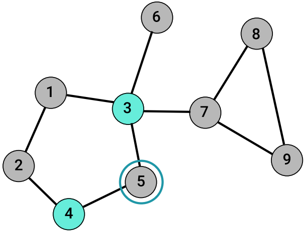
Graph Propagation

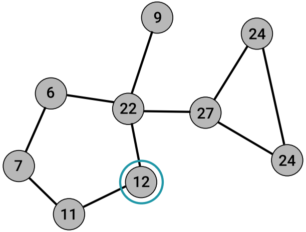
Simple graph representation (set $w = 1$): $y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j$



Graph Propagation

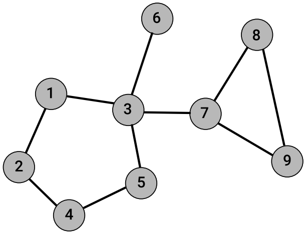
Simple graph representation (set $w = 1$): $y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j$

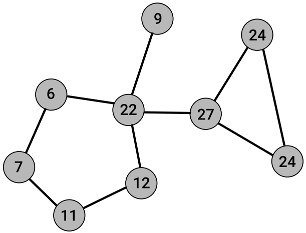




Graph Propagation

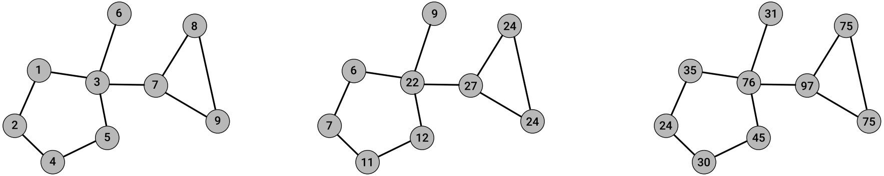
Simple graph representation (set $w = 1$): $y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j$





Graph Propagation

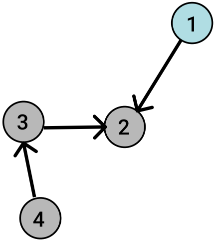
Simple graph propagation (set $w = 1$): $$y_i = x_i + \sum_{j \in \mathcal{N}_i} x_j$$



- if applied iteratively, it takes into account the structure

Simplest Graph Propagation

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

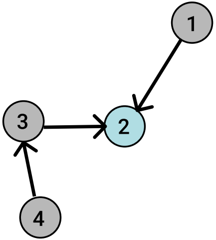


$$\boldsymbol{A} \in \mathbb{R}^{N \times N} \quad \boldsymbol{X} \in \mathbb{R}^{N} \quad \boldsymbol{Y} \in \mathbb{R}^{N}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 \\ \; \\ \; \\ \; \end{bmatrix}$$

Simplest Graph Propagation

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



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

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

$\boldsymbol{X} \in \mathbb{R}^{N}$

1
2
3
4

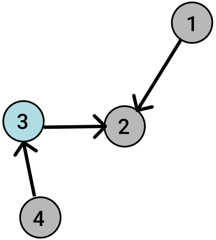
$=$

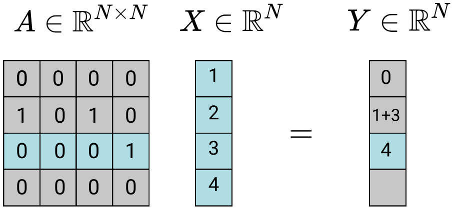
$\boldsymbol{Y} \in \mathbb{R}^{N}$

0
1+3

Simplest Graph Propagation

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

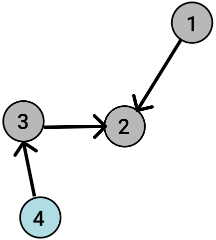




Simplest Graph Propagation

Bitdefender

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

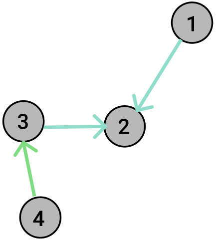


$$\boldsymbol{A} \in \mathbb{R}^{N \times N} \quad \boldsymbol{X} \in \mathbb{R}^{N} \quad \boldsymbol{Y} \in \mathbb{R}^{N}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1+3 \\ 4 \\ 0 \end{bmatrix}$$

Simplest Graph Propagation

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

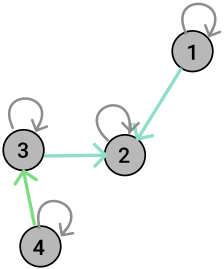


$$\boldsymbol{A} \in \mathbb{R}^{N \times N} \quad \boldsymbol{X} \in \mathbb{R}^{N \times D} \quad \boldsymbol{Y} \in \mathbb{R}^{N \times D}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ x_1 + x_3 \\ x_4 \\ 0 \end{bmatrix}$$

Simplest Graph Propagation

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



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

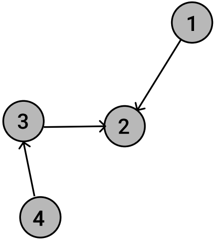
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_1 + x_2 + x_3 \\ x_3 + x_4 \\ x_4 \end{bmatrix}$$

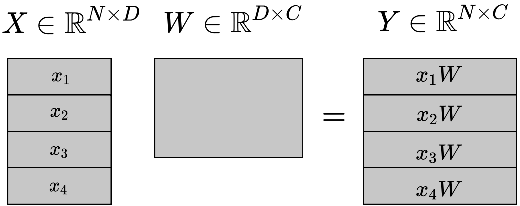
Simplest Graph Propagation

Bitdefender

To combine more complex representations:

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



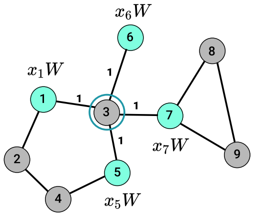


Simplest Graph Propagation

Bitdefender

To combine more complex representations:

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



The operations performed in the graph could be rewritten as:

$$Y = AXW$$

Iteratively, for more layers:

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

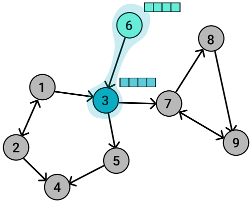
$$Y = A\sigma...A\sigma(AXW_1)W_2)..W_n$$

GNNs: Message Passing Framework - Send

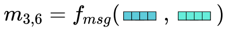
Bitdefender

Send Function

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



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

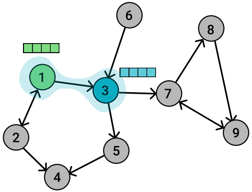


GNNs: Message Passing Framework - Send

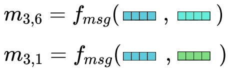
Bitdefender

Send Function

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



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

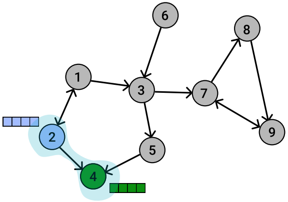


GNNs: Message Passing Framework - Send

Bitdefender

Send Function

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



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

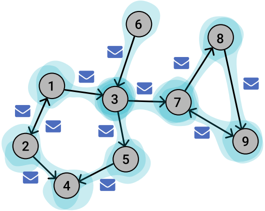
$$m_{3,6} = f_{msg}(\quad , \quad)$$

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

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

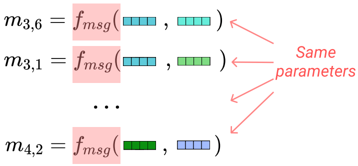
GNNs: Message Passing Framework - Send

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



Learnable funcion

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

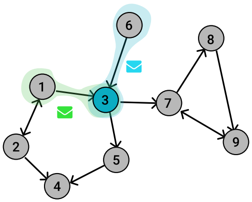


GNNs: Message Passing Framework - Aggregation

Bitdefender

Aggregation Function

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



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

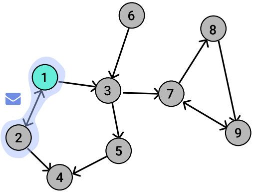
$h_3 = f_{agg}(\{$ ✉, ✉ $\})$

GNNs: Message Passing Framework - Aggregation

Bitdefender

Aggregation Function

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

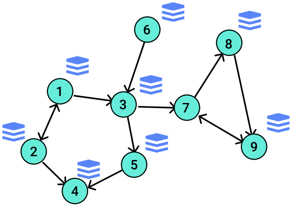


$$h_i = f_{agg}(\{m_{ij} | \forall j \in \mathcal{N}_i\})$$

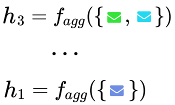
GNNs: Message Passing Framework - Aggregation

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



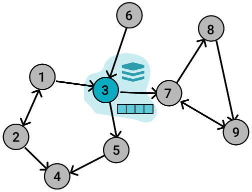
$$h_i = \overbrace{f_{agg}}^{\text{operator}} (\{m_{ij} | \forall j \in \mathcal{N}_i\}) \in \mathbb{R}^C$$



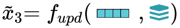
GNNs: Message Passing Framework - Update

Update Function

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



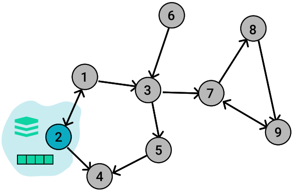
$$\tilde{x}_i = f_{upd}(x_i, h_i)$$



GNNs: Message Passing Framework - Update

Update Function

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



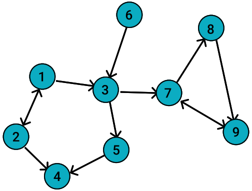
$$\tilde{x}_i = f_{upd}(x_i, h_i)$$

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

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

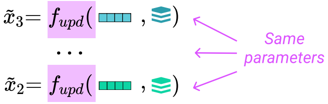
GNNs: Message Passing Framework - Update

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



Learnable function

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

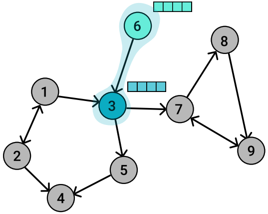


GNNs - Overview

Bitdefender

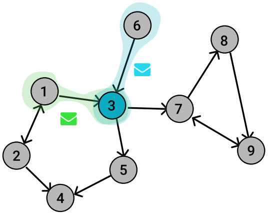
1. Send

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



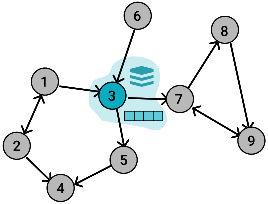
2. Aggregate

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



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

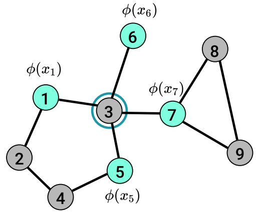
$$f_{upd}(x_i, \bigoplus_{\forall j \in \mathcal{N}_i} \{\alpha(x_i, x_j)\phi(x_j)\})$$

Message Passing

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

Graph Convolutional Network

Bitdefender



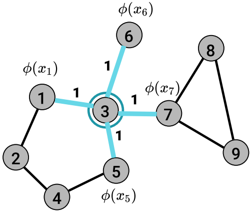
$$y_i = f_{upd}(x_i, \bigoplus_{\forall j \in \mathcal{N}_i} \{\phi(x_j)\})$$

- messages depend only on the source nodes

[10] *Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. ICLR 2017*

Graph Convolutional Network

Bitdefender

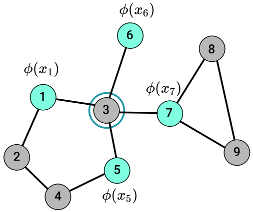


$$y_i = f_{upd}(x_i, \bigoplus_{\forall j \in \mathcal{N}_i} \{\phi(x_j)\})$$

- 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: $\frac{1}{deg(i)deg(j)}$

[10] *Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. ICLR 2017*

Graph Attention Network



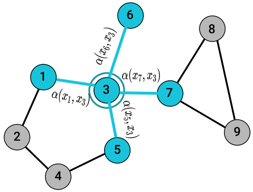
$$y_i = f_{upd}(x_i, \bigoplus_{\forall j \in \mathcal{N}_i} \{\alpha(x_i, x_j)\{\phi(x_j)\}\})$$

- messages depend only on the source nodes

[11] *Vaswani et. al. Attention is all you need. NeurIPS 2017*
[12] *Veličković et. al Graph attention networks. ICLR 2018*

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: $\alpha(x_i, x_j) \propto$ ReLU$(a^T[x_i W_1, x_j W_2])$

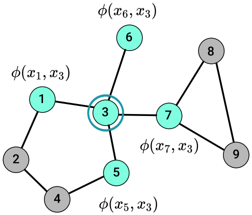
Self-Attention: $\alpha(x_i, x_j) \propto x_i W_1 (x_j W_2)^T$

- the model is able to learn the desired structure

---

[11] *Vaswani et. al. Attention is all you need. NeurIPS 2017*
[12] *Veličković et. al Graph attention networks. ICLR 2018*

Message Passing Neural Network



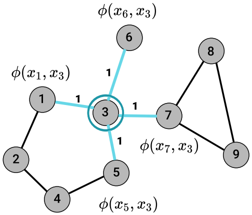
$$y_i = f_{upd}(x_i, \bigoplus_{\forall j \in \mathcal{N}_i} \{\phi(x_i, x_j)\})$$

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

[13] *Battaglia et. al. Interaction networks. NeurIPS 2016*
[14] *Gilmer et. al. Neural message passing for quantum chemistry. ICML 2017*

Message Passing Neural Network

Bitdefender



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

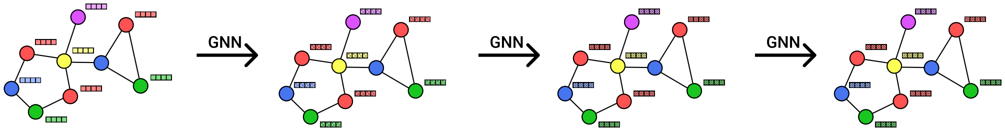
- messages depend on both source and destination
- if edge features are available, the message could also take them into account
- aggregation function is implemented as a sum/mean operation

[13] *Battaglia et. al. Interaction networks. NeurIPS 2016*
[14] *Gilmer et. al. Neural message passing for quantum chemistry. ICML 2017*

Multiple Layers

Bitdefender

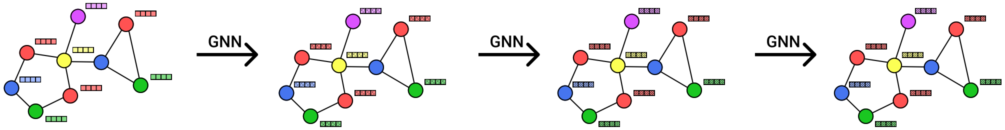
- for a more powerful representation, we can stack multiple layers



Multiple Layers

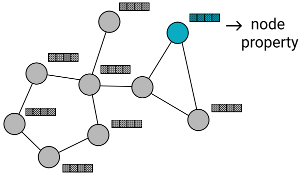
Bitdefender

- 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



- predict an output y_i from each node

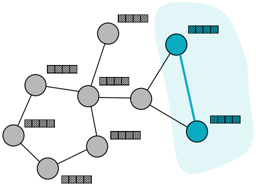
$$y_i = f_{output}(\tilde{x_i}) \quad \in \mathbb{R}^K$$

- 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



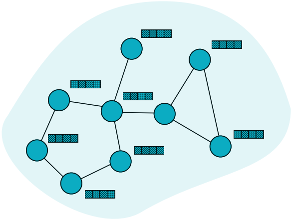
- predict an output y_{ij} from each pair of nodes

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

- 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



- predict a single output y for the whole graph

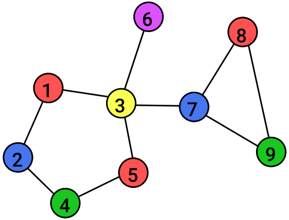
$$y = f_{readout}(\{\tilde{x}_i | \forall i \in \mathcal{V}\}) \quad \in \mathbb{R}^K$$

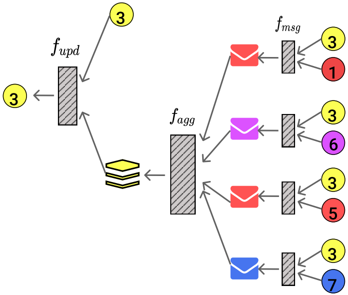
- $f_{readout}$ could be a simple order-invariant aggregator (e.g. sum, mean), or more complex graph pooling mechanisms
- the loss function is applied for each graph in the dataset

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

Learning

- 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**

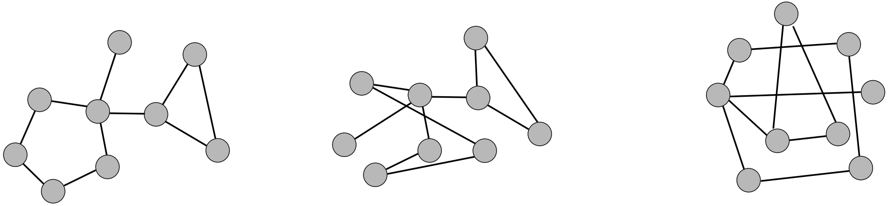




Expressive Power of GNNs

Bitdefender

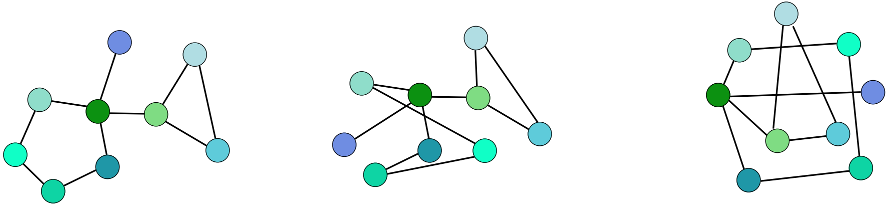
How many different graphs are in this image?



Expressive Power of GNNs

Bitdefender

How many different graphs are in this image?



Are Graph Neural Networks able to identify this?

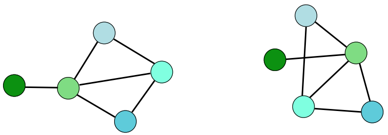
Isomorphism test

Bitdefender

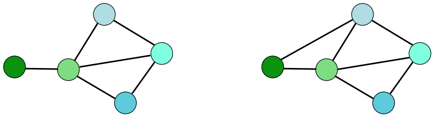
Graph isomorphism

Two graphs are **isomorphic** if and only if there exist a *mapping* from all nodes and all edges of a graph to the other or, more formally, if and only if there exists a permutation matrix P such that $PA_1P' = A_2$ and $PX_1 = X_2$.

ISOMORPH



NON - ISOMORPH



Isomorphism test

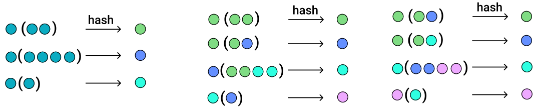
Bitdefender

Graph isomorphism

Two graphs are **isomorphic** if and only if there exist a *mapping* from all nodes and all edges of a graph to the other or, more formally, if and only if there exists a permutation matrix P such that $PA_1P' = A_2$ and $PX_1 = X_2$.

- no polynomial time algorithm is known to determine if two graphs are isomorphic
- **Weisfieler-Lehman Algorithm** (WL) is a powerful algorithm for isomorphism testing, but it still has cases when it goes wrong

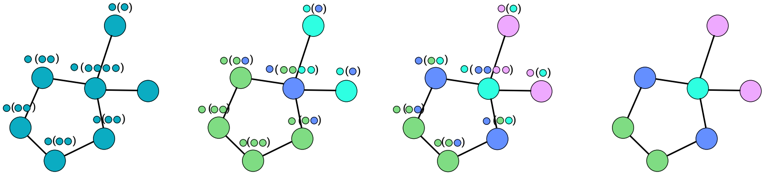
Weisfieler-Lehman Algorithm



Input: initial labels $l_0^0, l_1^0..l_N^0$
Output: final labels $l_0^T, l_1^T..l_N^T$

---

while *not reach a stable state* **do**
    for *each node* i **do**
        $l_i^t \leftarrow hash(l_i^{(t-1)}, \{\{l_j^{(t-1)}, \textbf{for } j \in \mathcal{N}_i\}\});$
    end
    $t \leftarrow t + 1;$
end

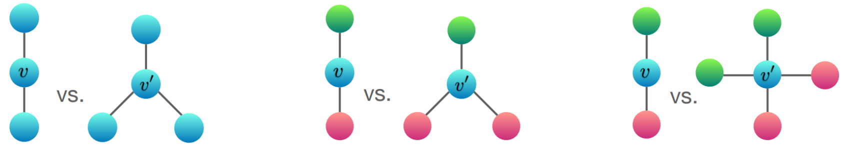


Expressive Power of GNNs

Bitdefender

Graphs expressive power *

A sufficient number of GNN layers maps any graphs G1 and G2 with input features from a countable universe are as powerful as the 1-WL test if f_{upd}, f_{agg} and $f_{readout}$ are injective.



(a) Mean and Max both fail (b) Max fails (c) Mean and Max both fail

*[15]: *Xu et. al. How powerful are graph neural networks? ICLR 2019*

Expressive Power of GNNs

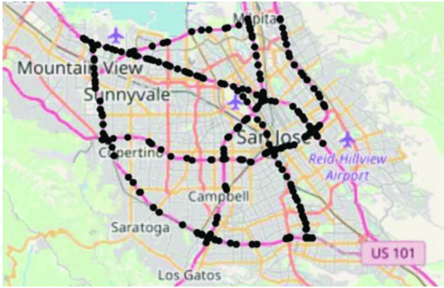
- Usually there is a trade-off between expressivity and generalization.
- We might want to sacrifice the isomorphism properties to be better aligned to the desired task (e.g. use min / max in some tasks) or to be able to learn more easily (e.g use attention).

GNN Application - Node Level

Bitdefender

Traffic forecasting *

For several road segments, predict the most likely traffic speed in the next H minutes.



Graph structure:

For each time step:

- *nodes:* traffic stations with traffic speed as features
- *edges:* depend on the location of the stations (e.g distance or topology of the streets)

For a time window we will have a series of graphs, one for each time step.

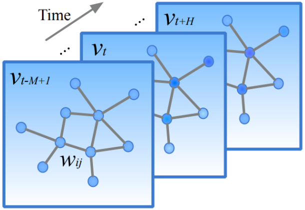
*[2]: *Yu et. al. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. IJCAI 2018*

GNN Application - Node Level

Bitdefender

Traffic forecasting

For several road segments, predict the most likely traffic speed in the next H minutes.



Graph model:

- spatial processing: for each time step, use a simple GCN (AXW) to process the nodes
- temporal processing: aggregate temporal information by using 1D Conv, independently for each node.

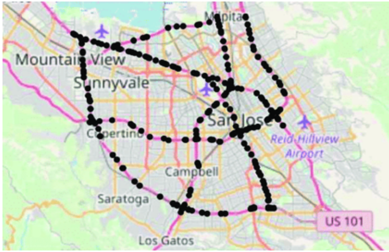
Combine: temporal-spatial-temporal + a Conv layer to reduce the temporal dimension. From each node predict the speed for the corresponding station.

GNN Application - Node Level

Bitdefender

Traffic forecasting

For several road segments, predict the most likely traffic speed in the next H minutes.



Why Graph Processing?

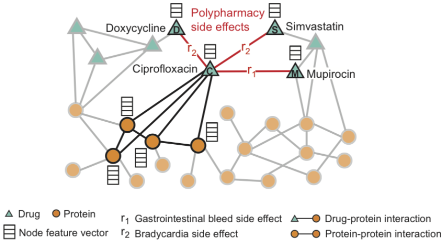
- The speed in one place is highly influenced by the traffic condition of near by roads segments.
- The model could better predict the traffic conditions if we take into account the whole roads network.

GNN Application - Edge Level

Bitdefender

Drug-Drug Interactions *

Predict if there are interactions between two drugs when administered simultaneously: can a change occur in the effects of one drug by the presence of another drug?



Graph structure:

For each time step:

- nodes: the drugs (no features)
- edges: are drawn between two drugs if we have information that those two drugs interact with each other

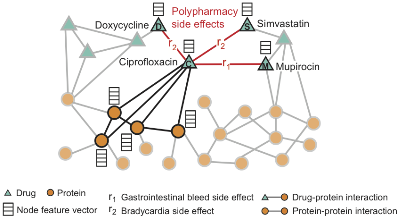
*[5]: *Huang et. al. Skipgnn: predicting molecular interactions with skip-graph networks. Sci Rep 10, 21092 (2020)*

GNN Application - Edge Level

Bitdefender

Drug-Drug Interactions

Predict if there are interactions between two drugs when administered simultaneously: can a change occur in the effects of one drug by the presence of another drug?



Graph model:

- GCNs (AXW) are applied to capture the relations between connected drugs.
- For two target nodes we concatenate their representation and predict the binary classification.

GNN Application - Edge Level

Bitdefender

Drug-Drug Interactions

Predict if there are interactions between two drugs when administered simultaneously: can a change occur in the effects of one drug by the presence of another drug?

Why Graph Processing?

- Motivated by a medical observation: two drugs could be similar (as side effects) if they behave in the same way when administered simultaneously with another drug.
- The GNN is able to encode the already discovered interactions between drugs and also the behavioral similarity between different drugs.
- The predicted interactions was supported by the medical literature.

GNN Application - Graph Level

Bitdefender

Action recognition *

Classify the action in the video. In general, the actions highly depend on the interactions happening in the scene.

Graph structure:

The graph structure is not explicitly provided in this case. One way to build it:

- nodes: objects / entities in the video.
- edges: represent similarity or interactions between objects.

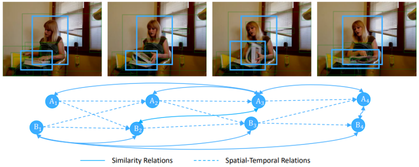
*[3]: *Wang and Gupta. Videos as space-time region graphs. ECCV 2018*

GNN Application - Graph Level

Bitdefender

Action recognition

Classify the action in the video. In general, the actions highly depend on the interactions happening in the scene.



Graph structure:

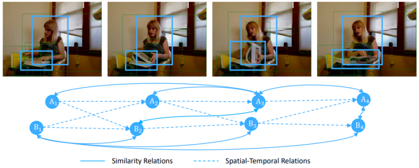
- nodes are extracted using a pre-trained object detector
- two types of graphs could be built:
  1. *similarity graph*: edges between all the nodes, regardless of the time step
  2. *spatial graph*: for two time steps $(t, t+1)$ draw an edge if $IoU > threshold$. Similar for $(t, t-1)$ pairs.

GNN Application - Graph Level

Bitdefender

Action recognition

Classify the action in the video.



Graph model:

– a GCN (AXW) is applied for each type of graph structure and the results are fused.

– to obtain a representation at the graph level (for the whole video) we aggregate all the nodes in the graph.

GNN Application - Graph Level

Bitdefender

Action recognition

Classify the action in the video.

Why Graph Processing?

– the GCNs is able to capture the correlation between objects understanding how they interact with each other and to “track” the objects across the temporal dimension

GNN Application - Vision Approaches

Bitdefender

Other Approaches

- the nodes shouldn't necessary be associated with objects. There are approaches that use pixels or patches as nodes and propagate information between them. RSTG [16], ViT [17]

- we can dynamically predict the salient regions, for cases when we do not have access to an object detector or we do not know what type of information to store in each node of the graph. [18]

---

[16] *Nicolicioiu, Duta, Leordeanu. Recurrent space-time graph neural networks NeurIPS 2019*
[17] *Dosovitskiy et. al. An Image is Worth 16x16 Words: Transformers for Image Recognition ICLR 2021*
[18] *Duta, Nicolicioiu, and Leordeanu. Dynamic regions graph neural networks for spatio-temporal reasoning. NeurIPS - ORLR Workshop 2020*

Graph Neural Networks - Resources

This lecture was influenced by several great resources about Graph Neural Networks. For a more in depth understanding of Graph Neural Networks and other related areas, please take a look:

- Michael Bronstein, *Geometric deep learning, from Euclid to drug design* Link
- Petar Veličković, *Theoretical Foundations of Graph Neural Networks* Link
- Jure Leskovec, *CS224W: Machine Learning with Graphs* Link
- William L. Hamilton, *Graph Representation Learning Book* Link
- Razvan Pascanu, *GraphNets - Lecture at TMLSS (Transylvanian Machine Learning Summer School)*
- Michael Bronstein, *Graph Deep Learning Blog* Link

Thank You!

Next week: Graph Neural Networks Part 2

Iulia Duta
iduta@bitdefender.com

Andrei Nicolicioiu
anicolicioiu@bitdefender.com

UNIVERSITATEA DIN BUCUREȘTI
VIRTUTE ET SAPIENTIA

Bitdefender

May 2021

References I

[1] Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein.
Fake news detection on social media using geometric deep learning.
arXiv preprint arXiv:1902.06673, 2019.

[2] Bing Yu, Haoteng Yin, and Zhanxing Zhu.
Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting.
pages 3634–3640. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[3] Xiaolong Wang and Abhinav Gupta.
Videos as space-time region graphs.
In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 399–417, 2018.

[4] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein.
Geometric deep learning on graphs and manifolds using mixture model cnns.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.

References II

[5] Kexin Huang, Cao Xiao, Lucas M Glass, Marinka Zitnik, and Jimeng Sun.
Skipgnn: predicting molecular interactions with skip-graph networks.
Scientific reports, 10(1):1–16, 2020.

[6] Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho.
Discovering symbolic models from deep learning with inductive biases.
In *Advances in Neural Information Processing Systems*, 2020.

[7] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli.
Graph matching networks for learning the similarity of graph structured objects.
In *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 3835–3845. PMLR, 09–15 Jun 2019.

[8] Petar Veličković, Lars Buesing, Matthew Overlan, Razvan Pascanu, Oriol Vinyals, and Charles Blundell.
Pointer graph networks.
In *Advances in Neural Information Processing Systems*, volume 33, pages 2232–2244. Curran Associates, Inc., 2020.

References III

Bitdefender

[9] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec.
Graph convolutional neural networks for web-scale recommender systems.
In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

[10] Thomas N. Kipf and Max Welling.
Semi-supervised classification with graph convolutional networks.
In *International Conference on Learning Representations (ICLR)*, 2017.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin.
Attention is all you need.
In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[12] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio.
Graph attention networks.
In *International Conference on Learning Representations*, 2018.

References IV

[13] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al.
Interaction networks for learning about objects, relations and physics.
In *Advances in neural information processing systems*, pages 4502–4510, 2016.

[14] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl.
Neural message passing for quantum chemistry.
In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272, 2017.

[15] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka.
How powerful are graph neural networks?
In *International Conference on Learning Representations*, 2019.

[16] Andrei Nicolicioiu, Iulia Duta, and Marius Leordeanu.
Recurrent space-time graph neural networks.
In *Advances in Neural Information Processing Systems 32*, pages 12838–12850. Curran Associates, Inc., 2019.

References V

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby.
An image is worth 16x16 words: Transformers for image recognition at scale.
In *International Conference on Learning Representations*, 2021.

[18] Iulia Duta, Andrei Nicolicioiu, and Marius Leordeanu.
Dynamic regions graph neural networks for spatio-temporal reasoning.
NeurIPS 2020 Workshop on Object Representations for Learning and Reasoning, 2020.