Graph Neural Networks

Introduction - Part 2

Andrei Nicolicioiu

Iulia Duta

UNIVERSITATEA DIN BUCUREȘTI

VIRTUTE ET SAPIENTIA

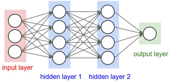
Bitdefender®

May 2021

Choose your model

UNSTRUCTURED

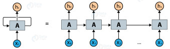
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	Iris setosa
4.9	3.0	1.4	0.2	Iris setosa
7.0	3.2	4.7	1.4	Iris versicolor
6.4	3.2	4.5	1.5	Iris versicolor
6.3	3.3	6.0	2.5	Iris virginica
5.8	3.3	6.0	2.5	Iris virginica



MLP

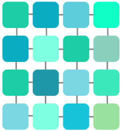
SEQUENTIAL

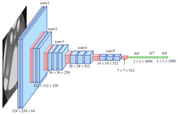
Have a nice day! :)



RNN

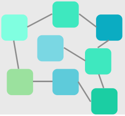
GRID

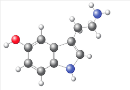


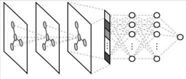


CNN

RELATIONAL STRUCTURE

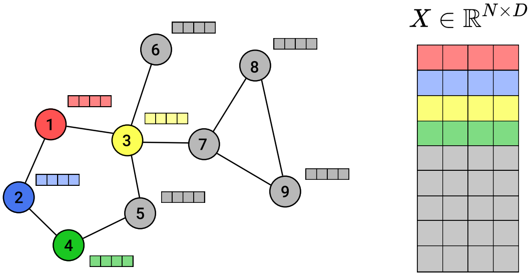






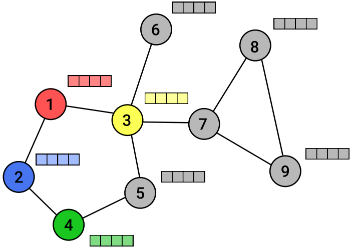
GNN

Graph Data

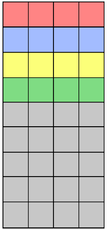


- all the nodes $x_i \in \mathbb{R}^D$ are stacked into a matrix $X \in \mathbb{R}^{N \times D}$
- each row corresponds to a node $x_i \in \mathbb{R}^D$

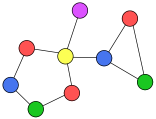
Graph Data



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



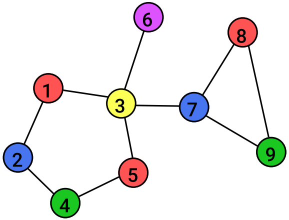
- all the nodes $x_i \in \mathbb{R}^D$ are stacked into a matrix $X \in \mathbb{R}^{N \times D}$
- each row corresponds to a node $x_i \in \mathbb{R}^D$

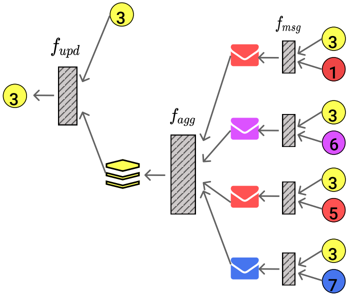
$f\Big($  $\Big) = Y$

Learning

Bitdefender

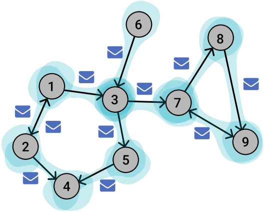
- the output of a GNN for a node i is obtained by applying a **sequence of operations** on the initial nodes
- all the operations along the sequence should be **differentiable**





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

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

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

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

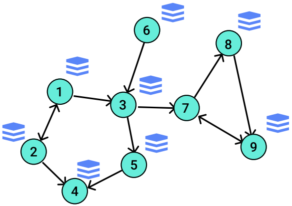
Same parameters

$$\dots$$

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

GNNs: Message Passing Framework - Aggregation

Bitdefender



- aggregate incoming messages with the function f_{agg}:
  - usualy not learnable: eg. sum, mean, max, min
  - learnable: e.g. LSTM
- it should be **invariant to the order** of the nodes and should **allow a variable number** of messages

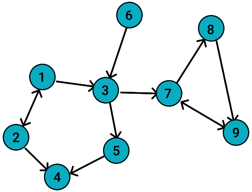
$$h_i = \overbrace{f_{agg}}^{\text{operator}}(\{m_{ij} | \forall j \in \mathcal{N}_i\}) \in \mathbb{R}^C$$

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

$$\dots$$

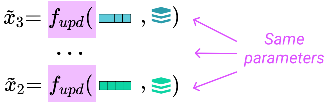
$$h_1 = f_{agg}(\{\text{✉}\})$$

GNNs: Message Passing Framework - Update



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

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

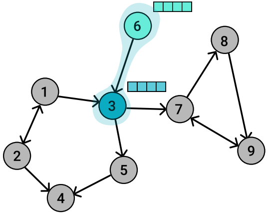


GNNs - Overview

Bitdefender

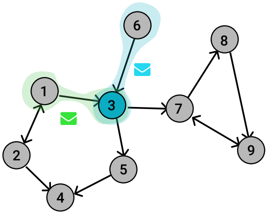
1. Send

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



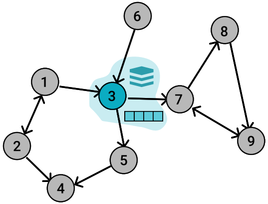
2. Aggregate

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



3. Update

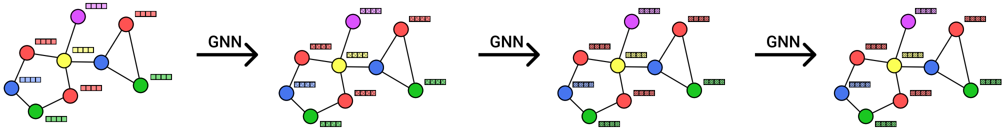
$$\tilde{x}_i = f_{upd}(x_i, H_i)$$



Multiple Layers

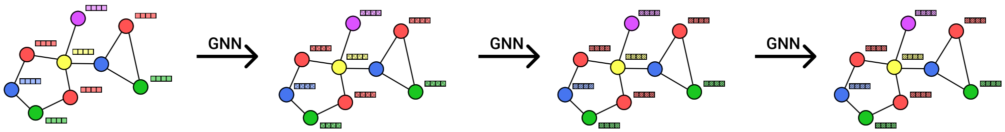
Bitdefender

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



Multiple Layers

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



RECEPTIVE FIELD:

Application: Fake News Identification

Goal: determine if a Tweet links to a fake news article

[redacted]

New paper from Brain Zurich and Berlin!

We try a conv and attention free vision architecture: MLP-Mixer (arxiv.org/abs/2105.01601)

Application: Fake News Identification

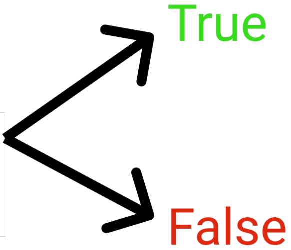
Bitdefender

Goal: determine if a Tweet links to a fake news article

[redacted]

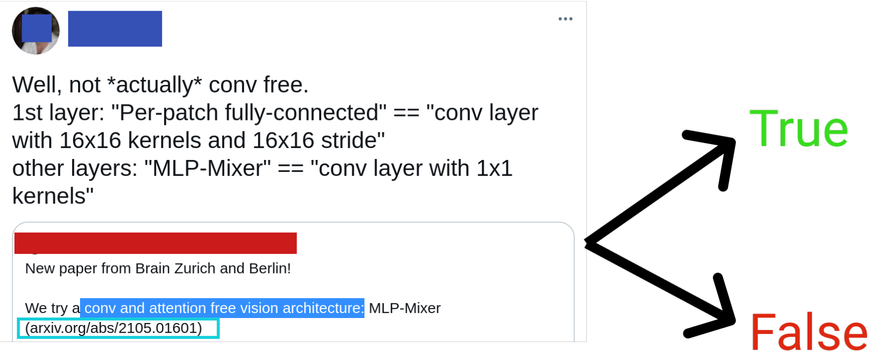
New paper from Brain Zurich and Berlin!

We try a conv and attention free vision architecture: MLP-Mixer (arxiv.org/abs/2105.01601)



Application: Fake News Identification

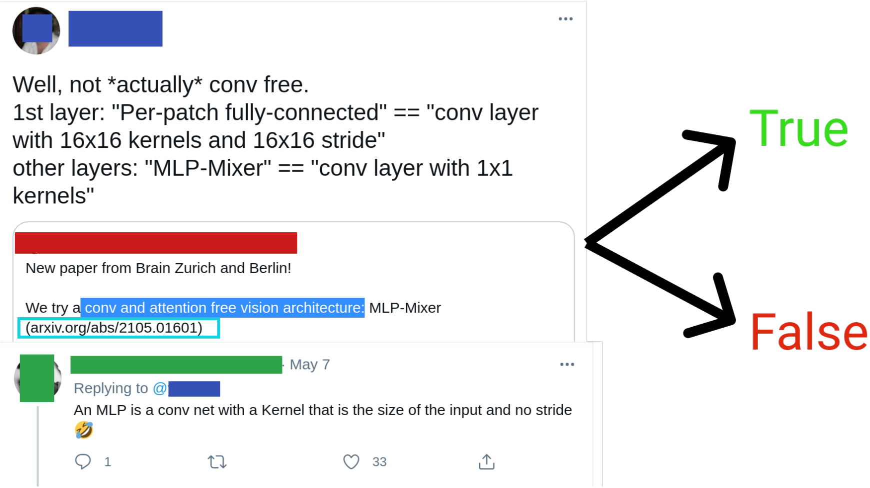
Goal: determine if a Tweet links to a fake news article



Application: Fake News Identification

Bitdefender

Goal: determine if a Tweet links to a fake news article



Application: Fake News Identification

Bitdefender

Challenges:

- understanding news requires knowledge of political / social context
- often written in bad faith to appear real
- highly nuanced

[1][1]: Vosoughi et. al. The spread of true and false news online. Science (2018).

Application: Fake News Identification

Bitdefender

Challenges:

- understanding news requires knowledge of political / social context
- often written in bad faith to appear real
- highly nuanced

News Spread

"Falsehood diffused significantly *farther, faster, deeper, and more broadly* than the truth"[1]

[1] [1]: Vosoughi et. al. The spread of true and false news online. Science (2018).

Application: Fake News Identification

Bitdefender

Challenges:

- understanding news requires knowledge of political / social context
- often written in bad faith to appear real
- highly nuanced

News Spread

"Falsehood diffused significantly *farther, faster, deeper, and more broadly* than the truth"[1]

Idea

- Analyse the news diffusion patterns with GNNs.

[1] [1]: Vosoughi et. al. The spread of true and false news online. Science (2018).

Application: Fake News Identification

Bitdefender

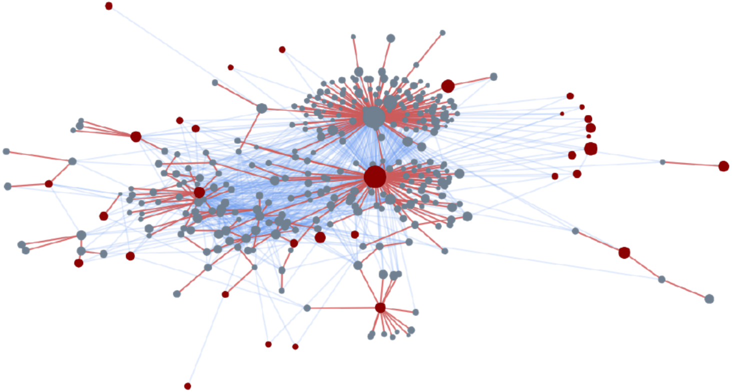
- gather stories classified by fact-checking orgs like Snopes, PolitiFact
- for each story form a graph of all the tweets and retweets mentioning it
- edges are follow relations or retweet relations

[2]: Monti et. al. Fake news detection on social media using geometric deep learning (2019).

Application: Fake News Identification

Bitdefender

- gather stories classified by fact-checking orgs like Snopes, PolitiFact
- for each story form a graph of all the tweets and retweets mentioning it
- edges are follow relations or retweet relations

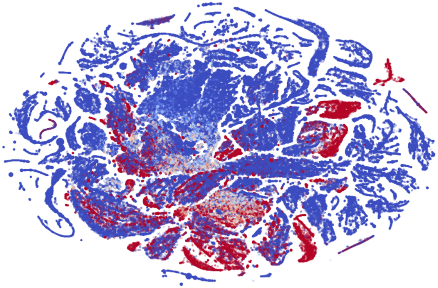


[2]: Monti et. al. Fake news detection on social media using geometric deep learning (2019).

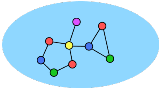
Application: Fake News Identification

Bitdefender

- apply standard GNN model
- node features:
  - User profile (geolocalization, language, embedding of self-description, date of account creation)
  - Network and spreading (No. of followers, timestamps, No. of replies, quotes, favorites and retweets for the source tweet)
  - Content (embeddings of tweet text).
    - Surprising: not that relevant!

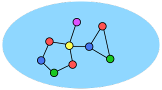


When to use GNNs?



- What are the cases where it is beneficial to use GNNs? 

When to use GNNs?



- What are the cases where it is beneficial to use GNNs?
- What design choices should be made for a specific task?
    - Do we want sum or max in the aggregation?
    - Should we share parameters between layers?
    - Should we use distances or order information when we have them?

When to use GNNs?

Bitdefender

Usual deep Learning approach:

- learn end-to-end $f(X)$ from data with the specific model f (MLP, CNN, RNN etc.)
- each model is appropriate in certain cases

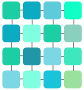
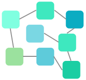
When to use GNNs?

Usual deep Learning approach:

- learn end-to-end $f(X)$ from data with the specific model f (MLP, CNN, RNN etc.)
- each model is appropriate in certain cases

Inductive biases

An inductive bias allows a learning algorithm to prioritize one solution over another, independent of the observed data.

	UNSTRUCTURED	SEQUENTIAL	GRID	COMPLEX RELATIONAL STRUCTURE
Structure				
Model	MLP	RNN	CNN	GNN
Inductive Bias	Weak	Sequentiality	Locality	Strong relational bias

Relational Reasoning

Bitdefender

Relational Reasoning

Manipulating *structured* data, that consists in multiple **entities** that establish various **relations** between them.

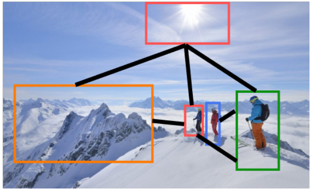
Relational Reasoning

Relational Reasoning

Manipulating *structured* data, that consists in multiple **entities** that establish various **relations** between them.

From some perspective, relational reasoning could be appealing.
For example, a visual scene could be seen as:

- an image / a grid of points
- a set of objects with multiple relations between them



Relational Inductive Biases

Bitdefender

Inductive biases
An inductive bias allows a learning algorithm to prioritize one solution over another, independent of the observed data.

Relational inductive biases in GNN:

- explicit factorisation into nodes, each corresponding to an entity
- explicit modeling of pairwise relations between nodes
- flexibility in establishing different connectivity
- order invariant

[3] Battaglia et. al. Relational inductive biases, deep learning, and graph networks. 2018

When to use a GNN?

Bitdefender

GNNs could be appropiate if:

- there exist entities and relations in the data
    - explicit: social networks, molecules
    - implicit: visual scenes, environments...
- the relational processing is beneficial

Shortest path Problem

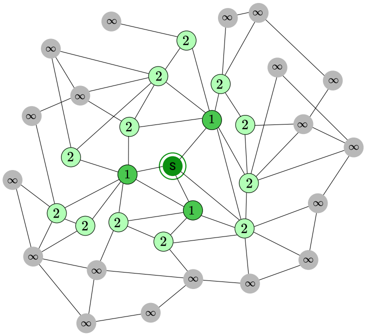
Bitdefender

- Lets analyse a purely reasoning problem of finding the shortest path in a graph.
  - Can GNNs solve this problem and how sample efficient are they?

Shortest path Problem

Bitdefender

- Lets analyse a purely reasoning problem of finding the shortest path in a graph.
    - Can GNNs solve this problem and how sample efficient are they?



Shortest path Problem

Bitdefender

- Lets analyse a purely reasoning problem of finding the shortest path in a graph.
    - Can GNNs solve this problem and how sample efficient are they?

GNN Method

for layer k in 1 .. K **do**
  for node i in $\mathcal{V}$ **do**
    $$x_i^k = f_{upd}\{x_i^{k-1}, \underset{\forall j \in \mathcal{N}_i}{f_{agg}} \{f_{msg}(x_i^{k-1}, x_j^{k-1})\}$$
  end for
end for

Shortest path Problem

Bitdefender

- Lets analyse a purely reasoning problem of finding the shortest path in a graph.
  - Can GNNs solve this problem and how sample efficient are they?

GNN Method	Bellman-Ford Algorithm

for layer k in 1 .. K **do**
  for node i in $\mathcal{V}$ **do**
    $x_i^k = f_{upd}\{x_i^{k-1}, \underset{\forall j \in \mathcal{N}_i}{f_{agg}} \{f_{msg}(x_i^{k-1}, x_j^{k-1})\}$
  end for
end for
	for iter k in 1 .. K **do**
	  for node i in $\mathcal{V}$ **do**
	    $d[k][i] = \underset{\forall j}{min}\{d[k-1][j] + cost(i,j)\}$
	  end for
	end for

Shortest path Problem

Bitdefender

- Lets analyse a purely reasoning problem of finding the shortest path in a graph.
  - Can GNNs solve this problem and how sample efficient are they?

GNN Method	Bellman-Ford Algorithm
for layer k in 1 .. K **do**	**for** iter k in 1 .. K **do**
for node i in $\mathcal{V}$ **do**	**for** node i in $\mathcal{V}$ **do**
$x_i^k = f_{upd}\{x_i^{k-1}, \underset{\forall j \in \mathcal{N}_i}{f_{agg}} \{f_{msg}(x_i^{k-1}, x_j^{k-1})\}$	$d[k][i] = \underset{\forall j}{min}\{ d[k-1][j] + cost(i,j)\}$
end for	**end for**
end for	**end for**

Shortest path Problem

Bitdefender

- Lets analyse a purely reasoning problem of finding the shortest path in a graph.
  - Can GNNs solve this problem and how sample efficient are they?

GNN Method	Bellman-Ford Algorithm

for layer k in 1 .. K **do**
    for node i in $\mathcal{V}$ **do**
        $x_i^k = f_{upd}\{x_i^{k-1}, \underset{\forall j \in \mathcal{N}_i}{f_{agg}} \{ f_{msg}(x_i^{k-1}, x_j^{k-1}) \}$
    end for
end for
	for iter k in 1 .. K **do**
	    for node i in $\mathcal{V}$ **do**
	        $d[k][i] = \underset{\forall j}{min}\{ d[k-1][j] + cost(i, j) \}$
	    end for
	end for

Shortest path Problem

Bitdefender

- Lets analyse a purely reasoning problem of finding the shortest path in a graph.
    - Can GNNs solve this problem and how sample efficient are they?

GNN Method	Bellman-Ford Algorithm

for layer k in 1 .. K **do**
    for node i in $\mathcal{V}$ **do**
        $x_i^k = f_{upd}\{x_i^{k-1}, \underset{\forall j \in \mathcal{N}_i}{f_{agg}} \{f_{msg}(x_i^{k-1}, x_j^{k-1})\}$
    end for
end for
	for iter k in 1 .. K **do**
	    for node i in $\mathcal{V}$ **do**
	        $d[k][i] = \underset{\forall j}{min} \{d[k-1][j] + cost(i, j)\}$
	    end for
	end for

Task Alignment

Bitdefender

- if the GNN learns to simulate the update step in the Dynamic Problem, it will solve the problem
- if the operation is easy to lean, then the GNN can easily solve the problem
- if both are true, we say that the GNN is *well aligned* with the task

[4] Xu et. al. What Can Neural Networks Reason About? ICLR 2020

Task Alignment

Bitdefender

- if the GNN learns to simulate the update step in the Dynamic Problem, it will solve the problem
- if the operation is easy to lean, then the GNN can easily solve the problem
- if both are true, we say that the GNN is *well aligned* with the task

Alignment

We say that a model is **aligned** with a task, if by replacing some parts of the model with some ideal operations we would solve the task. If the parts can *easily learn* the ideal operations, we say that it is **well aligned** with the task.

Generally:

- If a model is well aligned with a task, it will lean it easily (it has low sample complexity).

[4] Xu et. al. What Can Neural Networks Reason About? ICLR 2020

Task Alignment

GNN Method	Bellman-Ford Algorithm

for layer k in 1 .. K **do**
  for node i in $\mathcal{V}$ **do**
    $x_i^k = f_{upd}\{x_i^{k-1}, \underset{\forall j \in \mathcal{N}_i}{f_{agg}}\{f_{msg}(x_i^{k-1}, x_j^{k-1})\}$
  end for
end for
	for iter k in 1 .. K **do**
	  for node i in $\mathcal{V}$ **do**
	    $d[k][i] = \underset{\forall j}{min}\{d[k-1][j] + cost(i,j)\}$
	  end for
	end for

- What decision can we take to have the GNN "more aligned"?

Task Alignment

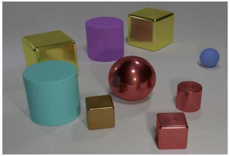
Bitdefender

GNN Method	Bellman-Ford Algorithm

for layer k in 1 .. K **do**
  for node i in $\mathcal{V}$ **do**
    $x_i^k = f_{upd}\{x_i^{k-1}, \underset{\forall j \in \mathcal{N}_i}{f_{agg}}\{f_{msg}(x_i^{k-1}, x_j^{k-1})\}$
  end for
end for
	for iter k in 1 .. K **do**
	  for node i in $\mathcal{V}$ **do**
	    $d[k][i] = \underset{\forall j}{min}\{d[k-1][j] + cost(i,j)\}$
	  end for
	end for

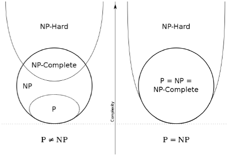
- What decision can we take to have the GNN "more aligned"?
    - use **min** as an aggregator function
    - **share** the parameters between layers
- Is $\tilde{x}_i = \text{MLP}([x_1, x_2, ...x_N])$ well aligned?
    - it is less aligned than the GNN functions
    - it has to learn to create node pairs and then it has to select the minimum between

Alignment

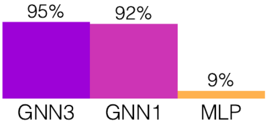


Relational argmax
What are the colors of the furthest pair of objects?

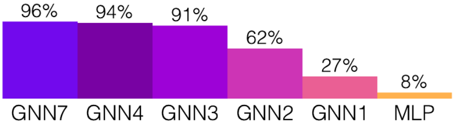
Dynamic programming
What is the cost to defeat monster X by following the optimal path?



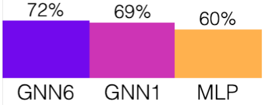
NP-hard problem
Subset sum: Is there a subset that sums to 0?



Relational argmax



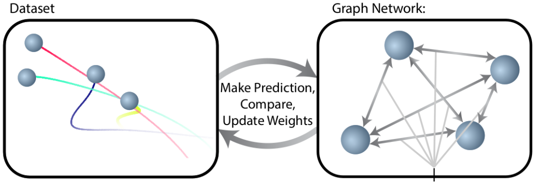
Dynamic Programming



NP - hard problem (random = 50%)

Alignment: Physical Particles

Bitdefender



GNNs	Analogy to Newtonian Mechanics
Nodes	Particles
Pair of nodes	Two interacting particles (i,j)
Send Function: f_{msg}	Compute force F_{ij}
Aggregate Function: f_{msg}	Sum into net force $F_{net,i}$
Update Function: f_{msg}	Compute acceleration $a_i = F_{net,i}/m_i$

[5]: Cranmer et. al. Discovering Symbolic Models from Deep Learning with Inductive Biases. Neurips 2020.

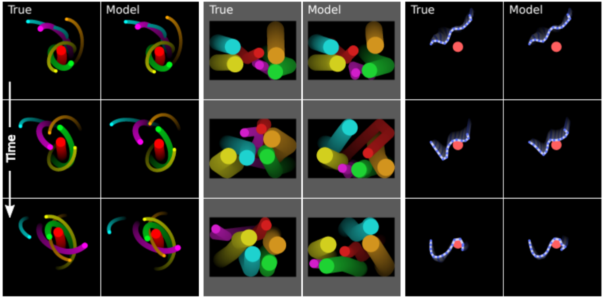
When to use a GNN?

- Apply GNNs on tasks that are well aligned with this model
  - dynamic programming
  - relational reasoning
- Apply GNNs when relational processing is beneficial
  - explicit entities and relations: social networks, molecules
  - implicit entities and relations: visual scenes, environments...
- Try to design your GNN to be as aligned as possible to your problem

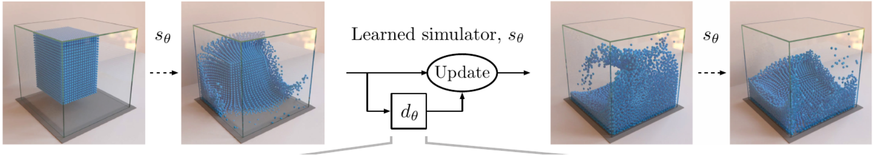
Application: Physical Particle Interactions

Bitdefender

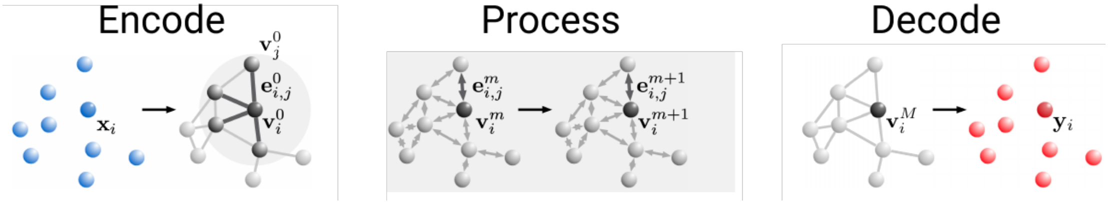
[6] Battaglia et. al. NeurIPS 2016



[7] Gonzalez et al. ICML 2020



Application: Physical Particle Interactions

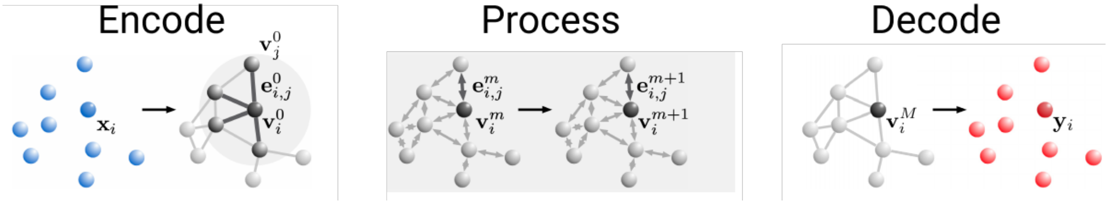


Encoder:

- each node corresponds to a particle
- link top-k nearest neighbors
- Node features:
    - position and velocity
    - particle type

[7] Gonzalez et al. ICML 2020

Application: Physical Particle Interactions



Process:

- use 10 GNN layers
- local propagation based on neighbourhood

Decoder:

- predict next step attributes
- train based on node level loss

[7] Gonzalez et al. ICML 2020

Application: Physical Particle Interactions

Bitdefender

Observations:

- the method is traned for next step predictions but at test time is unrolled for thousand of steps
- GNN method could generalise to 34 times more nodes at test time
  - because the interactions to nearest neighbours
- relative positions are is better than global positions
  - underlying physical processes are invariant to spatial position,

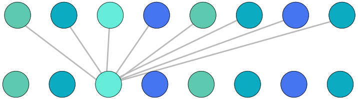
Overall:

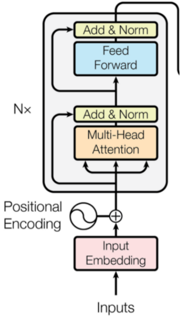
- GNN is aligned to the task
- the GNN has built in good relational biases
  - use local interactions
  - relative position for built in spatial invariance

Transformer

Bitdefender

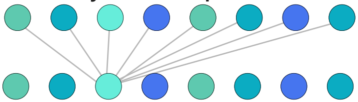
Task: analyse a sequence of words. $X = x_1, x_2, ..., x_N$.



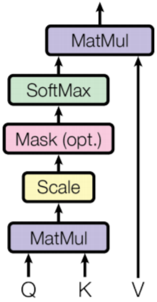


Transformer

Task: analyse a sequence of words. $X = x_1, x_2, ..., x_N$.



Scaled Dot-Product Attention



Self - Attention

- Process a sequence in multiple layers
- Each element attends to all other elements in the previous layer

$$Y = \text{softmax}(\frac{QK^T}{\sqrt{d}})V$$

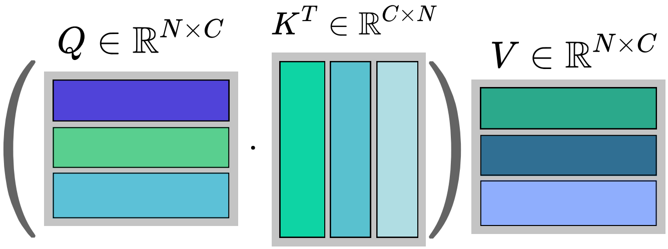
- where $Q = XW_q, K = XW_k, V = XW_v$

Transformer

Self-attention

$$Y = \text{softmax}(\frac{QK^T}{\sqrt{d}})V$$

where $Q = XW_q, K = XW_k, V = XW_v$



Transformer

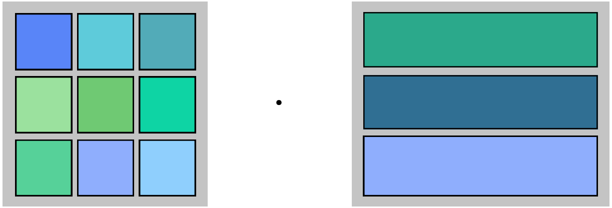
Bitdefender

Self-attention

$$Y = \text{softmax}(\frac{QK^T}{\sqrt{d}})V$$

where $Q = XW_q, K = XW_k, V = XW_v$

$A \in \mathbb{R}^{N \times N}$ $\qquad$ $V \in \mathbb{R}^{N \times C}$



Transformer

Self-attention

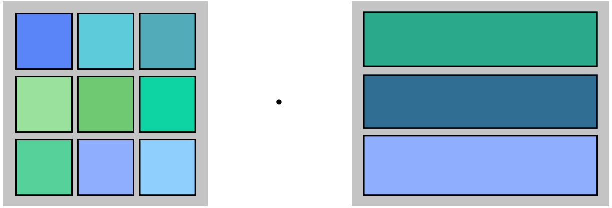
$$Y = \underbrace{\text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)}_{\boxed{A}} V$$

GCN

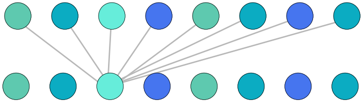
$$Y = \sigma(\boxed{A}\,\boxed{XW})$$

where $Q = XW_q$, $K = XW_k$, $\boxed{V = XW_v}$

$A \in \mathbb{R}^{N \times N}$ $\quad$ $V \in \mathbb{R}^{N \times C}$

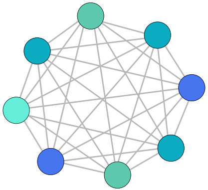


Transformer



$$Y = \frac{QK^T}{\sqrt{d}}V$$

$$y_i = \sum_{\forall j} \underbrace{\frac{1}{\sqrt{d}} \overbrace{(x_i W_q)}^{\text{Query}} \overbrace{(x_j W_k)^T}^{\text{Key}}}_{\alpha(x_i,x_j)} \overbrace{(x_j W_v)}^{\text{Value}}$$



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

$$\alpha(x_i, x_j) = \frac{1}{\sqrt{d}}(x_i W_q)^T (x_j W_k)$$

$$\phi(x_j) = x_j W_j$$

Transformer

Bitdefender

Transformers vs GNNs

Transformer is a special case of Graph Neural Networks where

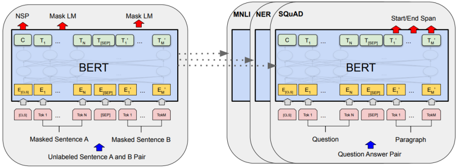
- all the nodes are connected
- pairwise messages are weighted by dot product attention

Transformer - NLP

Bitdefender

Transformers are now the standard model in NLP.

BERT [8]



Pre-training

Fine-Tuning

GPT-3 [9]

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
Translate English to French:     ← task description
cheese =>                        ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

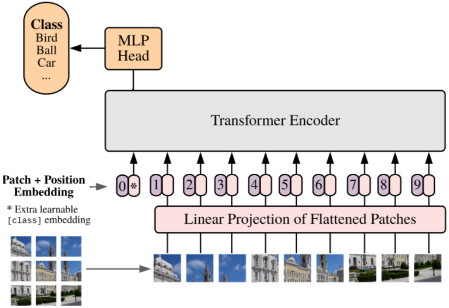
```
Translate English to French:     ← task description
sea otter => loutre de mer       ← example
cheese =>                        ← prompt
```

Transformer - Vision

Bitdefender

Transformers are becoming popular in CV.

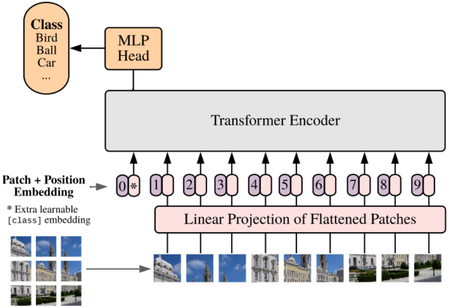
ViT [10]



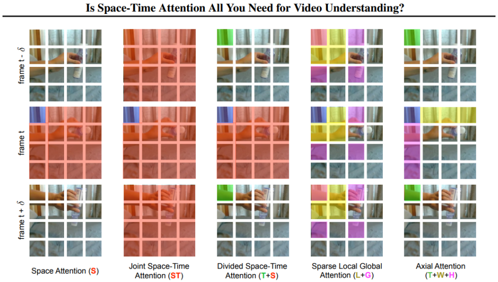
Transformer - Vision

Transformers are becoming popular in CV.

ViT [10]



TimeSformer [11]



GNN - Challenges: Scalability

Context:

- ML methods work with mini-batches where each element is independent

- in many *node level* graph tasks, the entire dataset forms a large graph where each node is connected to many other ones.

Problem:

- the whole graph is too big to fit into memory.

    - process independently the neighbourhood of each node

    - the neighbourhood could still grow exponentially:

Challenges: Scalability

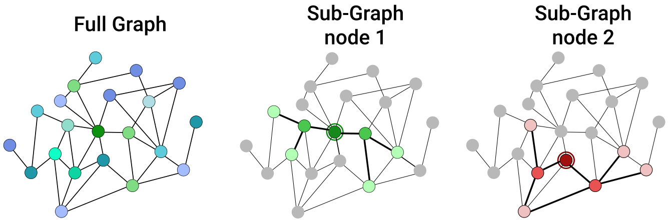
Bitdefender

Solution:

- sample [12],[13] the nodes, forming sub-graphs and apply the GNN over them

Benefits:

- can work with very large graphs
- the sampling acts as a regularizer, similar to dropout

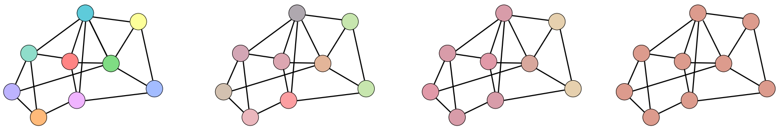


Challenges: Oversmoothing

Bitdefender

If we want node information from a K-order neighbourhood

- use K layers of Graph propagation
- usual problems
  - harder to optimize due to vanishing / exploding gradients
  - overfitting due to large number of parameters
- graph propagation problem: **oversmoothing**
  - graph propagation can be seens as "smoothing" the a node according to its neighbourhood
  - if we do many propagations, different nodes would become almost *indistinguishable*, hurting *node-level* tasks



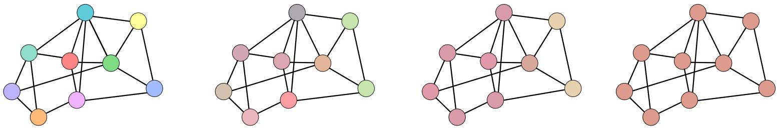
Challenges: Oversmoothing

Bitdefender

Oversmoothing
Nodes with similar structure in their neighbourhoods would end up indistinguishable, regardless of their initial features.

More often:

- when the graph is dense
- when using self-loop in the update function

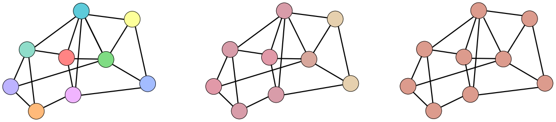


Oversmoothing: Solutions

Bitdefender

Solutions:

- **residual** Connections [14, 15]:
    - skip one or more layers
    - add the representations of a node from different layers $h_i^{k+1} \leftarrow h_i^{k+1} + h_i^k$
    - takes more into account the identity of each node

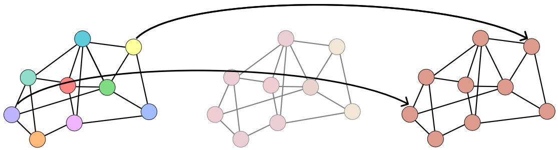


Oversmoothing: Solutions

Bitdefender

Solutions:

- **residual** Connections [14, 15]:
    - skip one or more layers
    - add the representations of a node from different layers $h_i^{k+1} \leftarrow h_i^{k+1} + h_i^k$
    - takes more into account the identity of each node

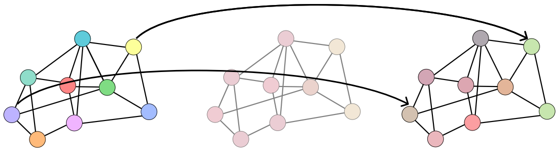


Oversmoothing: Solutions

Bitdefender

Solutions:

- **residual** Connections [14, 15]:
    - skip one or more layers
    - add the representations of a node from different layers $h_i^{k+1} \leftarrow h_i^{k+1} + h_i^k$
    - takes more into account the identity of each node



Oversmoothing: Solutions

Bitdefender

Solutions:

- make the graph more **sparse**: e.g apply dropout on edges [16]

- PairNorm[17]: add a **normalisation** term that encourages h_i^{t+1} and h_i^t to remain close while neighbouring nodes maximise their similarity and distant does minimise their similarity

Connections to PageRank

Bitdefender

Long range are obtained by stacking multiple layers: $A\sigma(A..\sigma(AXW_1)..W_{n-1})W_n$

Connections to PageRank

Bitdefender

Long range are obtained by stacking multiple layers: $A\sigma(A..\sigma(AXW_1)..W_{n-1})W_n$

Random Walk

- start in a node and randomly move to adjacency nodes.
- $W = I$ and $X \in \mathbb{R}^N$ a vector containing the probability of being in each node and A is the transition probability
- this arrives at the PageRank algorithm $X^{t+1} = AX^t$

Connections to Personalised PageRank

- PageRank converges to an Y that does not depend of the initial X
- this is related to the oversmoothing problem in the GNN
- in Personalised PageRank the initial starting point count more
  - at each step there is a chance α to go back to the initial state $X^{t+1} = (1-\alpha)AX^t + \alpha X^0$

Connections to Personalised PageRank

Bitdefender

- PageRank converges to an Y that does not depend of the initial X
- this is related to the oversmoothing problem in the GNN
- in Personalised PageRank the initial starting point count more
  - at each step there is a chance α to go back to the initial state $X^{t+1} = (1-\alpha)AX^t + \alpha X^0$
- we can use a similar formulation in our graph propagation to alleviate the oversmoothing
  - the residual connection could be seen as a non-probabilistic variant

Connections to Personalised PageRank

Bitdefender

How can it be used in GNNs?

- make a prediction independently at each node and propagate the answer [18]

$$X^1 = X^0 W$$
$$X^{t+1} = (1-\alpha) A X^t + \alpha X^0$$

Connections to Personalised PageRank

How can it be used in GNNs?

- make a prediction independently at each node and propagate the answer [18]

$$X^1 = X^0 W$$
$$X^{t+1} = (1-\alpha) A X^t + \alpha X^0$$

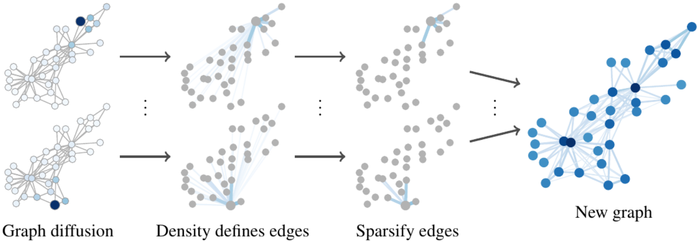
- this is somehow related to label propagation [19]

Connections to Personalised PageRank

Alternatively:

- compute Personalized Page Rank diffusion matrix S [20][21]
- sparsify the diffusion matrix
- and use it in a GCN

$$Y = \sigma(SXW)$$



Overview

- Graph Neural Network framework

Overview

Bitdefender

- Graph Neural Network framework
- application: fake news detection

Overview

Bitdefender

- Graph Neural Network framework
- application: fake news detection
- When to use GNNs?
  - relational inductive biases
  - alignment

Overview

Bitdefender

- Graph Neural Network framework
- application: fake news detection
- When to use GNNs?
    - relational inductive biases
    - alignment
- application: simulating particles

Overview

- Graph Neural Network framework
- application: fake news detection
- When to use GNNs?
    - relational inductive biases
    - alignment
- application: simulating particles
- Transformers are GNNs

Overview

Bitdefender

- Graph Neural Network framework
- application: fake news detection
- When to use GNNs?
  - relational inductive biases
  - alignment
- application: simulating particles
- Transformers are GNNs
- challenges: oversmoothing

Overview

Bitdefender

- Graph Neural Network framework
- application: fake news detection
- When to use GNNs?
    - relational inductive biases
    - alignment
- application: simulating particles
- Transformers are GNNs
- challenges: oversmoothing
- connections to PageRank

Graph Neural Networks - Resources

Bitdefender

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)*
- Xavier Bresson, *Convolutional Neural Networks on Graphs* Link
- Michael Bronstein, *Graph Deep Learning Blog* Link

Thank You!

Andrei Nicolicioiu
anicolicioiu@bitdefender.com

Iulia Duta
iduta@bitdefender.com

UNIVERSITATEA DIN BUCUREȘTI
VIRTUTE ET SAPIENTIA

Bitdefender

May 2021

References I

Bitdefender

[1] Soroush Vosoughi, Deb Roy, and Sinan Aral.
The spread of true and false news online.
Science, 359(6380):1146–1151, 2018.

[2] 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.

[3] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al.
Relational inductive biases, deep learning, and graph networks.
arXiv preprint arXiv:1806.01261, 2018.

[4] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken ichi Kawarabayashi, and Stefanie Jegelka.
What can neural networks reason about?
In *International Conference on Learning Representations*, 2020.

References II

Bitdefender

[5] 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.

[6] 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.

[7] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia.
Learning to simulate complex physics with graph networks.
In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 8459–8468, 2020.

References III

Bitdefender

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.
BERT: Pre-training of deep bidirectional transformers for language understanding.
In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, June 2019.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei.
Language models are few-shot learners.
In *Advances in Neural Information Processing Systems*, volume 33, 2020.

References IV

Bitdefender

[10] 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.

[11] Gedas Bertasius, Heng Wang, and Lorenzo Torresani.
Is space-time attention all you need for video understanding?
arXiv preprint arXiv:2102.05095, 2021.

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec.
Inductive representation learning on large graphs.
In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1024–1034. Curran Associates, Inc., 2017.

References V

Bitdefender

[13] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna.
GraphSAINT: Graph sampling based inductive learning method.
In *International Conference on Learning Representations*, 2020.

[14] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka.
Representation learning on graphs with jumping knowledge networks.
In *Proceedings of the 35th International Conference on Machine Learning, ICML, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5449–5458. PMLR, 2018.

[15] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem.
Deepgcns: Can gcns go as deep as cnns?
In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[16] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang.
Dropedge: Towards deep graph convolutional networks on node classification.
In *International Conference on Learning Representations*, 2020.

References VI

Bitdefender

[17] Lingxiao Zhao and Leman Akoglu.
Pairnorm: Tackling oversmoothing in gnns.
In *International Conference on Learning Representations*, 2020.

[18] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann.
Combining neural networks with personalized pagerank for classification on graphs.
In *International Conference on Learning Representations*, 2019.

[19] Hongwei Wang and Jure Leskovec.
Unifying graph convolutional neural networks and label propagation.
arXiv preprint arXiv:2002.06755, 2020.

[20] Johannes Klicpera, Stefan Weiß enberger, and Stephan Günnemann.
Diffusion improves graph learning.
In *Advances in Neural Information Processing Systems*, volume 32, 2019.

References VII

Bitdefender

[21] Emanuele Rossi, Fabrizio Frasca, Ben Chamberlain, Davide Eynard, Michael Bronstein, and Federico Monti.
Sign: Scalable inception graph neural networks.
arXiv preprint arXiv:2004.11198, 2020.