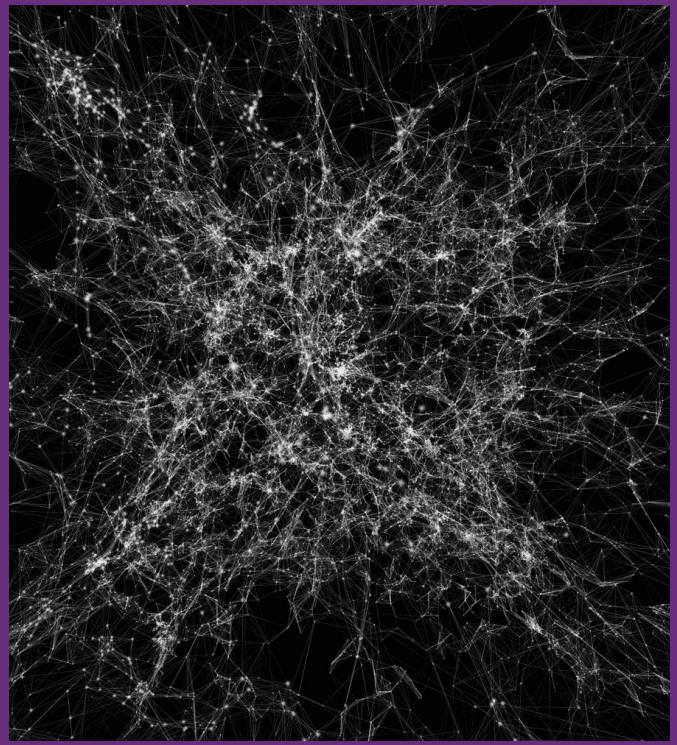


The Network Behind the Cosmic Web,
Barabasi et al, 2016

Mila

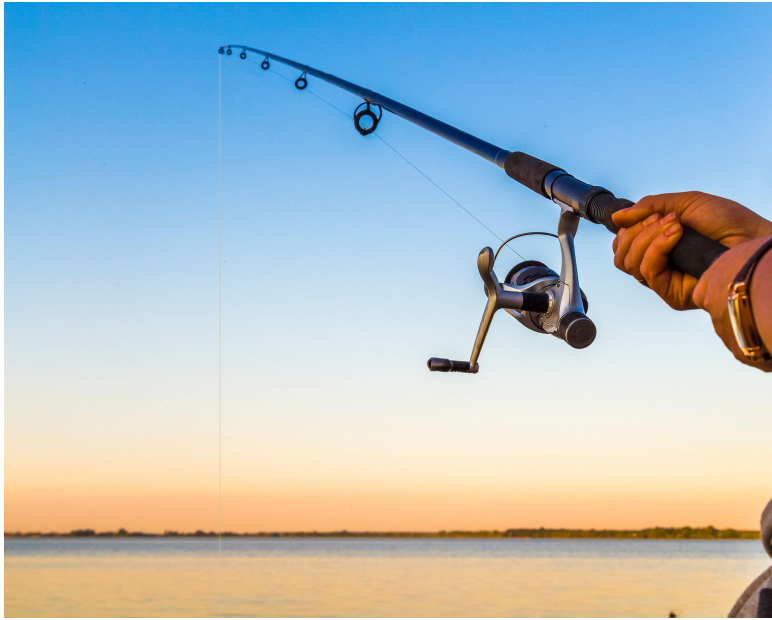
Reasoning with structure: Graph Neural Networks

17 August 2023

Andreea Deac

Overview

0. Motivation
1. Introduction to Graph Neural Networks
2. Graph (re)wiring
3. Geometric Deep Learning
4. Simulation/ Algorithmic reasoning



Graphs are everywhere

Natural inputs most commonly form a graph
- Molecules, social networks, road networks, brain connectome…
- Very rarely is an image / sequence the most faithful representation

Relations between entities form a graph
- Crucial aspect of input understanding

Reasoning processes form a (dynamic) graph
- At every step, conclusions based on a subset of nodes + their connections.
- These conclusions then connect with other nodes to form new conclusions
- Repeat “ad infinitum”?

Graph representation learning: an important component on the path to AGI
But graphs are still under-leveraged, especially across the sciences and in RL

Graphs are everywhere

Natural inputs most commonly form a graph
- Molecules, social networks, road networks, brain connectome…
- Very rarely is an image / sequence the most faithful representation

Relations betw
- Crucial a

Reasoning proc
- At every
- These conclusions then connect with other nodes to form new conclusions
- Repeat “ad infinitum”?

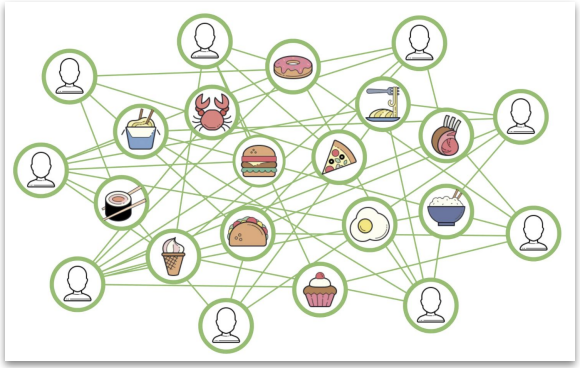
I aim to make **graphs** a first-class citizen in *natural* tasks!

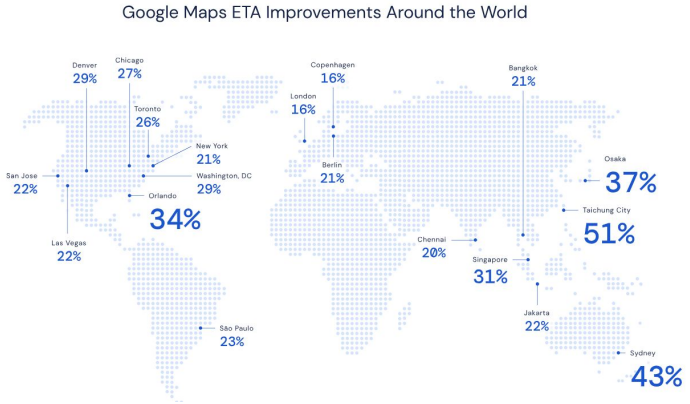
Take them from the background to the foreground, as **explicit information** to be leveraged

Graph representation learning: an important component on the path to AGI
But graphs are still under-leveraged, especially across the sciences and in RL

Applications of Graph Machine Learning

Food Discovery with Uber Eats: Using Graph Learning to Power Recommendations



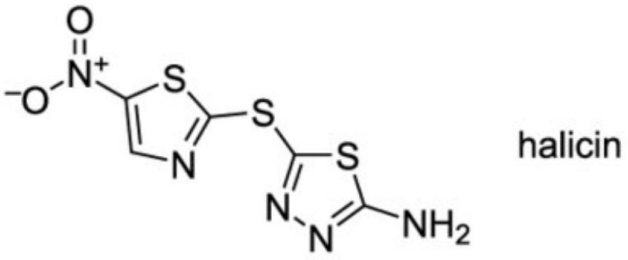


Traffic prediction with advanced Graph Neural Networks

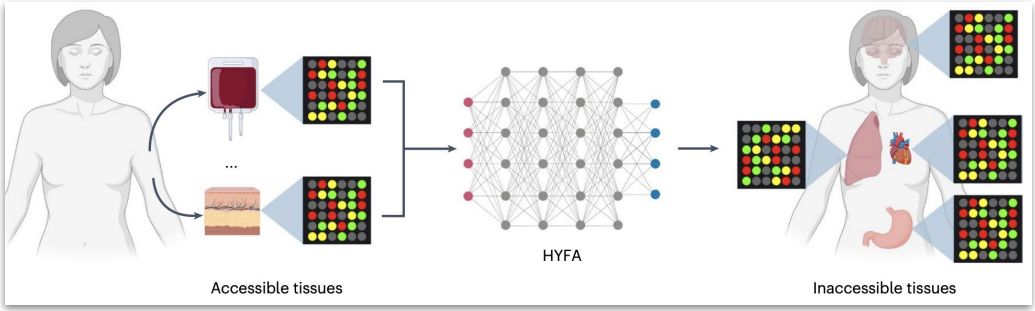
Mila

Applications of Graph Machine Learning

A Deep Learning Approach to Antibiotic Discovery, Stokes et al, Cell 2020



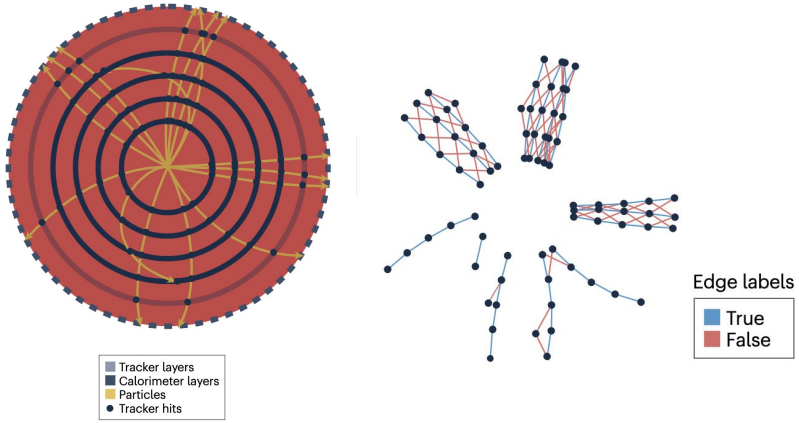
halicin

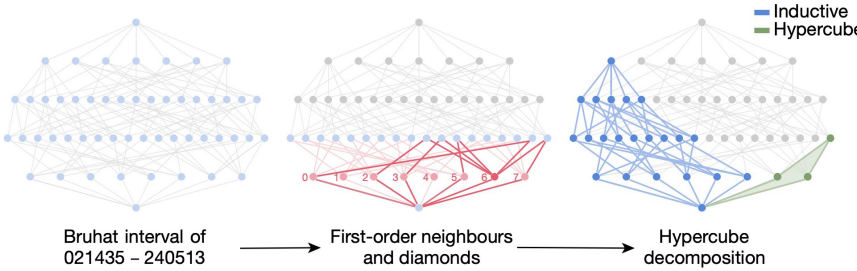


Hypergraph factorization for multi-tissue gene expression imputation, Vinas et al, Nat Mach Int 2023

Applications of Graph Machine Learning

Graph neural networks at the Large Hadron Collider, DeZoort et al, Nat Reviews Physics 2023



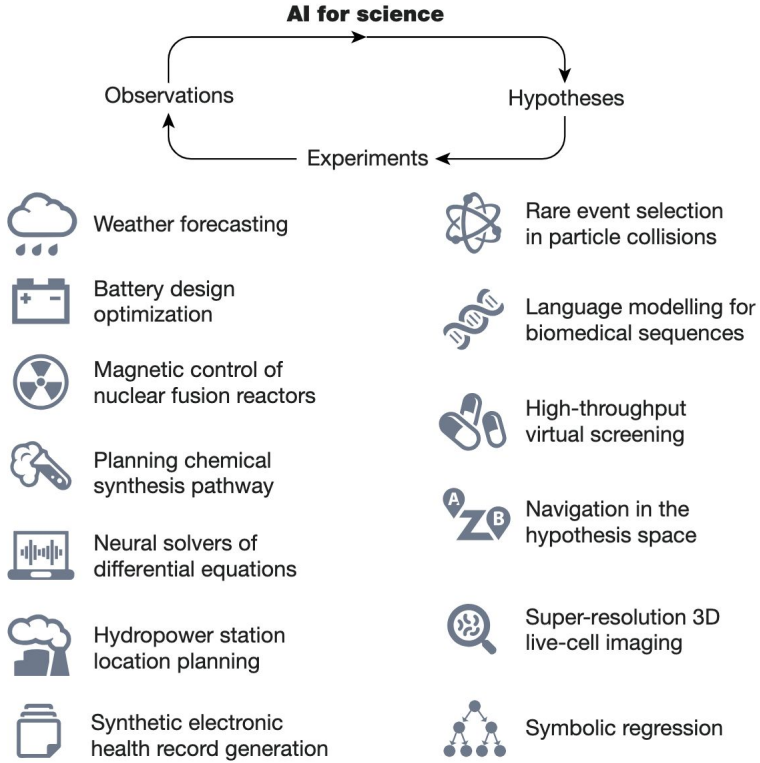


Advancing mathematics by guiding human intuition with AI, Davies et al, Nature 2021

Mila

And much more

Scientific discovery in the age of artificial intelligence, Wang et al, Nature 2023

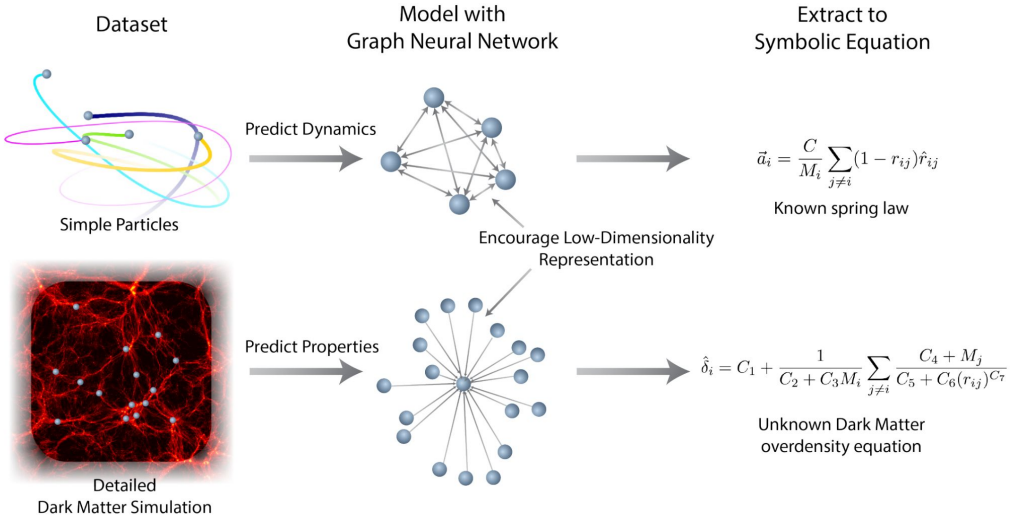


GNNs in Astrophysics

Graph is formed of:

- nodes: halos
- edges: i is connected to j if the distance between them is less than D

- train a Graph Neural Network to predict the smoothed overdensity δ_i at the location of the center of each halo
- apply symbolic regression to extract explicit physical relations
- recover known equations, but also a new analytic formula of the concentration of dark matter from the mass distribution of nearby cosmic structures



Discovering Symbolic Models from Deep Learning with Inductive Biases, Cranmer et al, NeurIPS 2020

GNNs in Astrophysics

1. Mangrove: Learning Galaxy Properties from Merger Trees
2. Rediscovering orbital mechanics with machine learning
3. New applications of Graph Neural Networks in Cosmology
4. Inferring halo masses with Graph Neural Networks
5. Galaxies and haloes on graph neural networks: Deep generative modelling scalar and vector quantities for intrinsic alignment
6. Unsupervised Resource Allocation with Graph Neural Networks

....

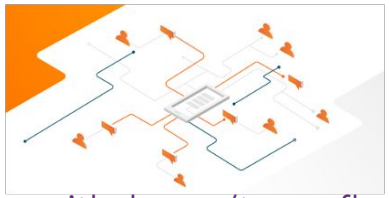
?. Scientific discovery in Astrophysics?

Available libraries

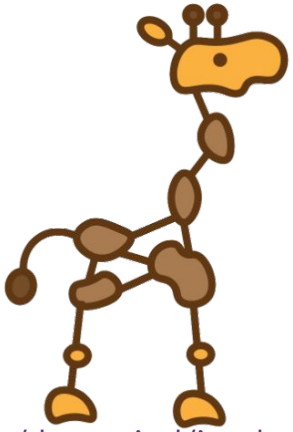
pyg.org

graphneural.network

dgl.ai



github.com/tensorflow/gnn

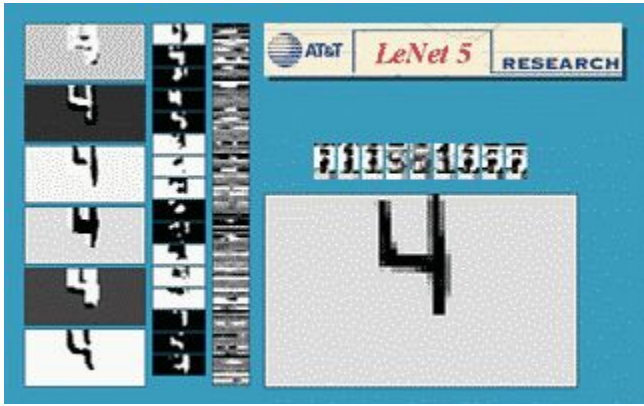


github.com/deepmind/jraph

Part I: Intro to GNNs

Convolutional Neural Networks

Neural networks on images ⟹ convolution

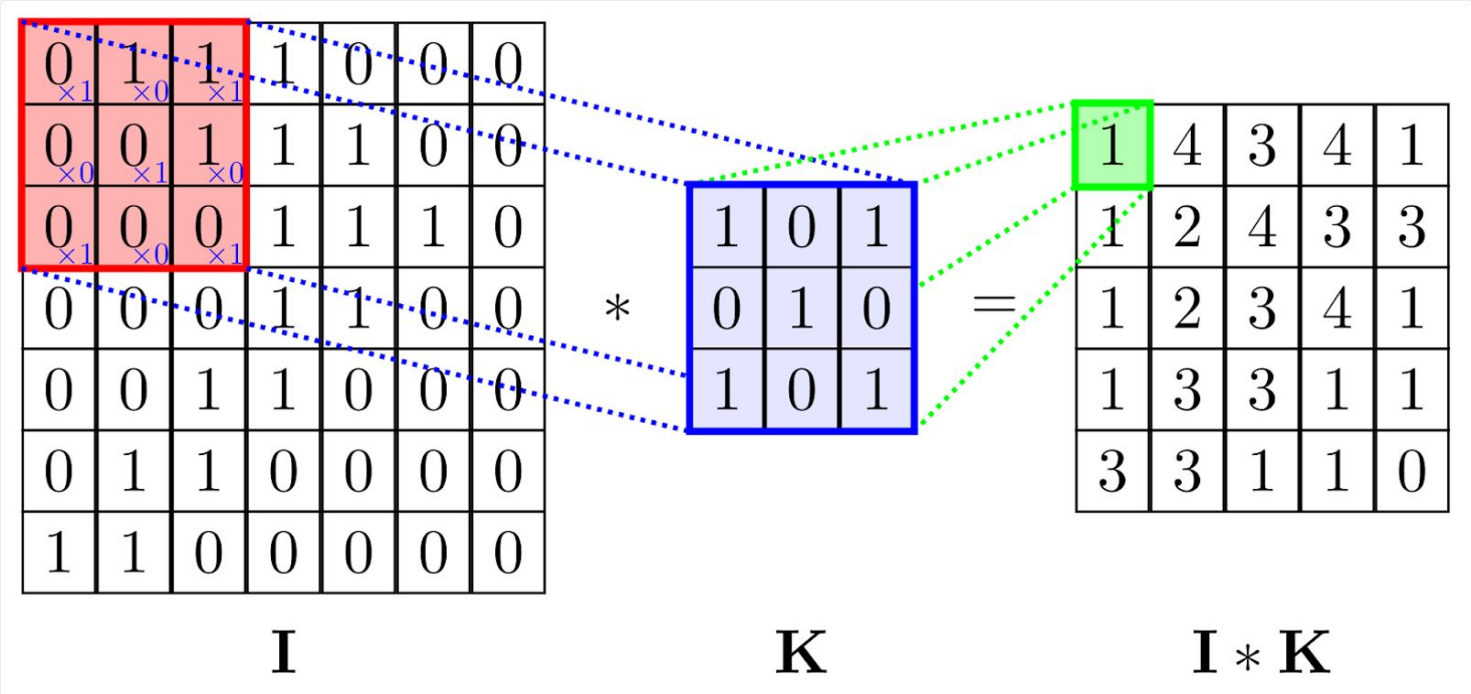


Convolutions have some very nice properties:

- The output changes predictably when the input is shifted
- Errors in one input pixel do not propagate to the entire output
- The computation is distributed across neighbourhoods

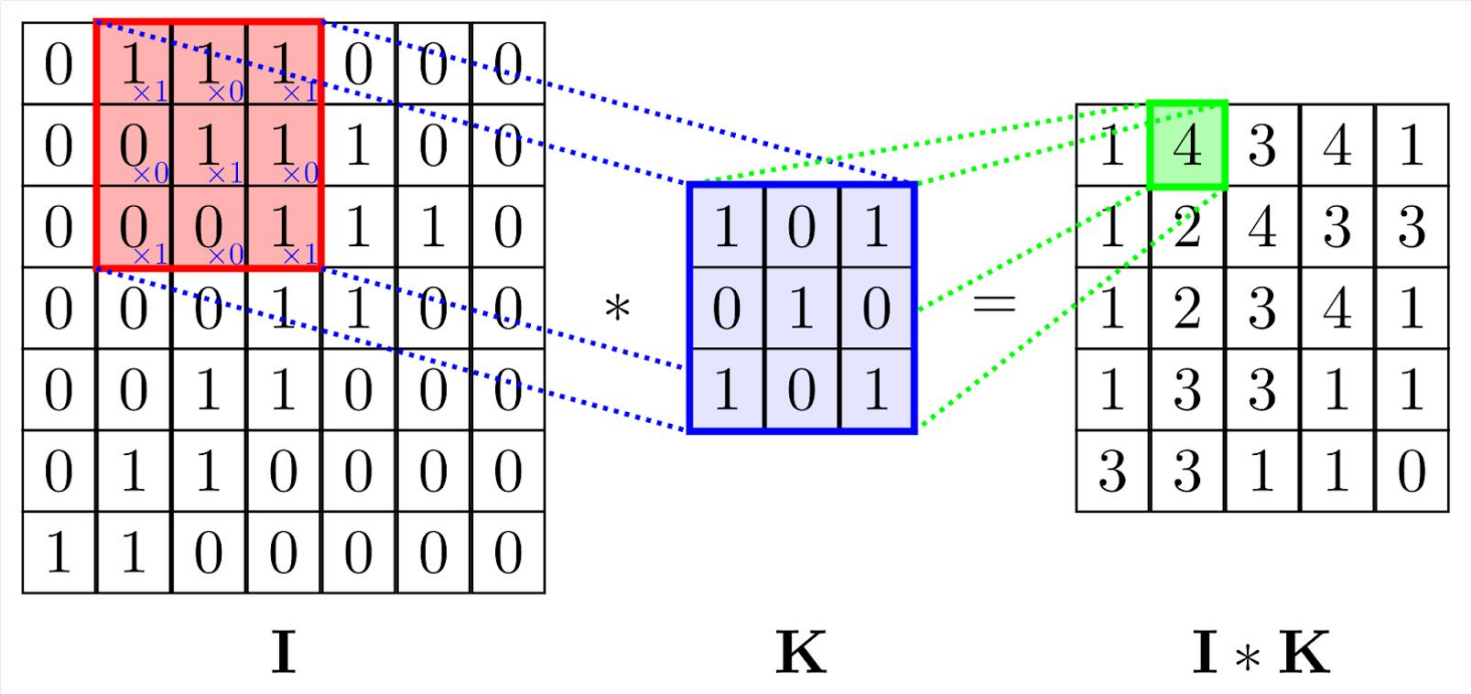
How do we formalise these concepts when moving from grids to graphs?

Convolutional Neural Networks



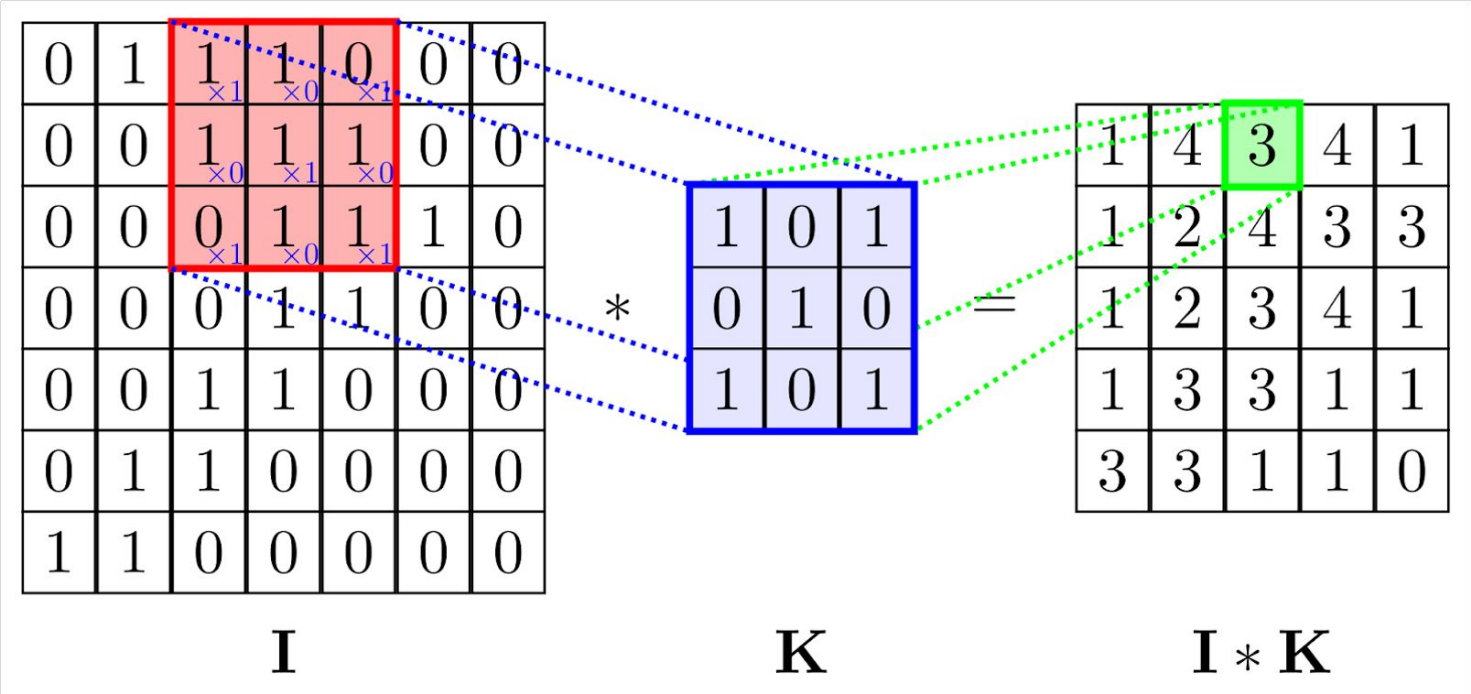
Mila

Convolutional Neural Networks



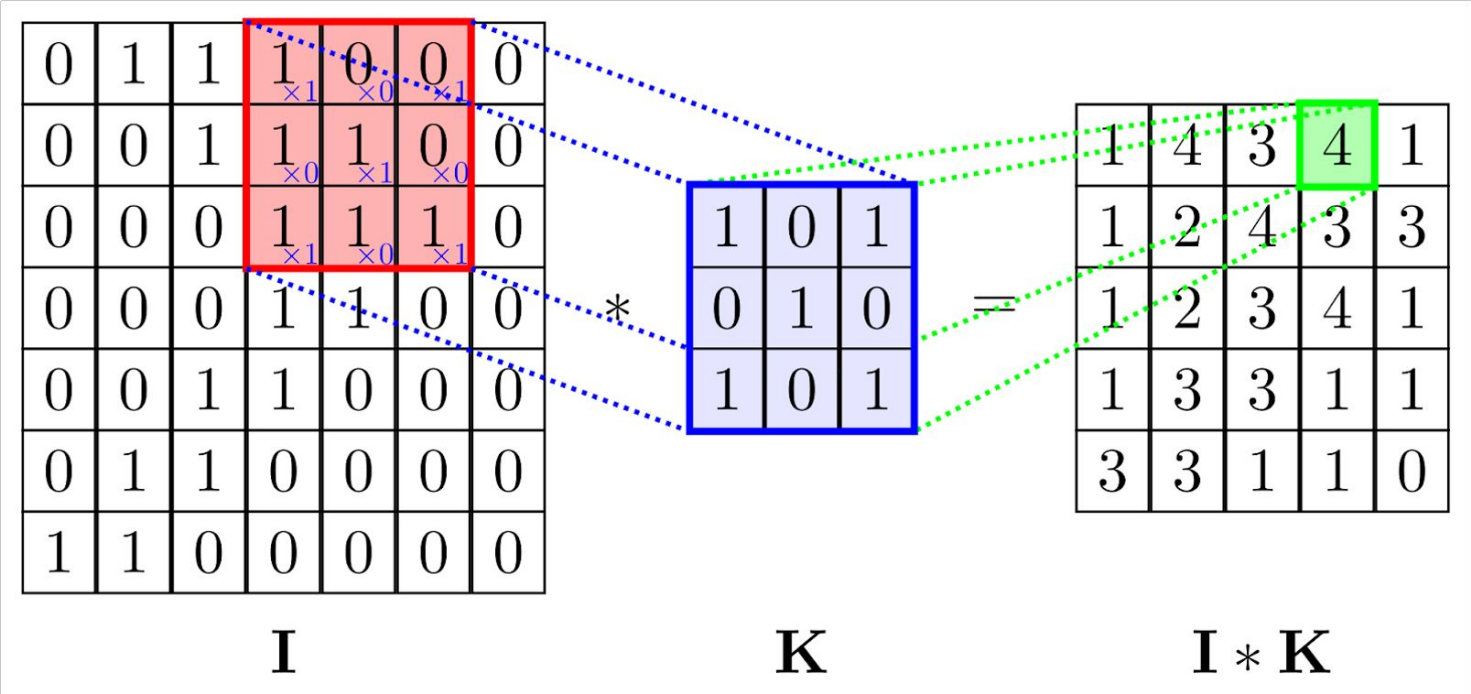
Mila

Convolutional Neural Networks



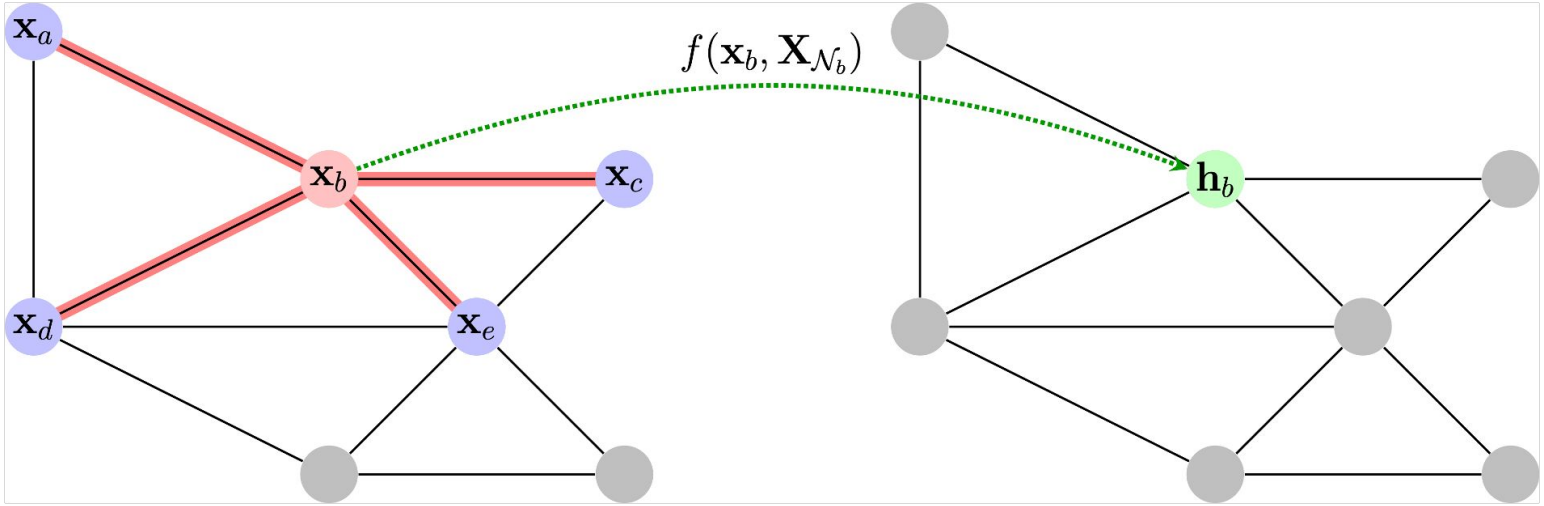
Mila

Convolutional Neural Networks



Mila

Graph Neural Networks



Mila

Graph formulation

Graphs: collections of objects (nodes) + interactions (edges) between them

Formally, a graph $\mathcal{G}=(\mathcal{V}, \mathcal{E})$ is a tuple of nodes ($\mathcal{V}$) and edges ($\mathcal{E}$).
Edges typically operate over pairs of nodes, i.e. $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$.
Depending on the context, nodes may be referred to as vertices, edges as links or relations.

We can represent edges with an adjacency matrix, $\mathbf{A} \in \mathbb{R}^{(|\mathcal{V}| \times |\mathcal{V}|)}$, such that $a_{ij} = 1$ if $(i,j) \in \mathcal{E}$ and 0 if $(i,j) \notin \mathcal{E}$

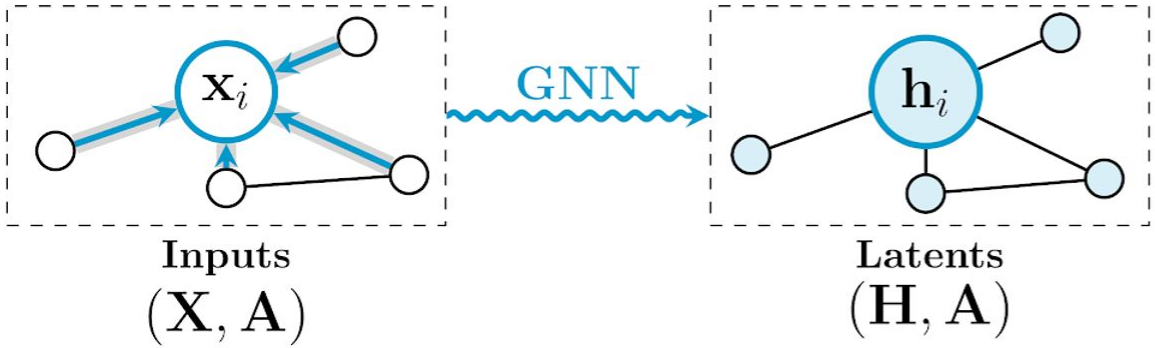
Let $\mathbf{x}_i \in \mathbb{R}^K$ be the features of node i. We can stack these features into a node feature matrix of shape $|\mathcal{V}| \times k$:

$$\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$$

That is, the i-th row of $\mathbf{X}$ corresponds to $\mathbf{x}_i$.

Thus, the input to a GNN is usually of the form $(\mathbf{X}, \mathbf{A})$.

Learning on graphs



Tasks on a graph: by scale

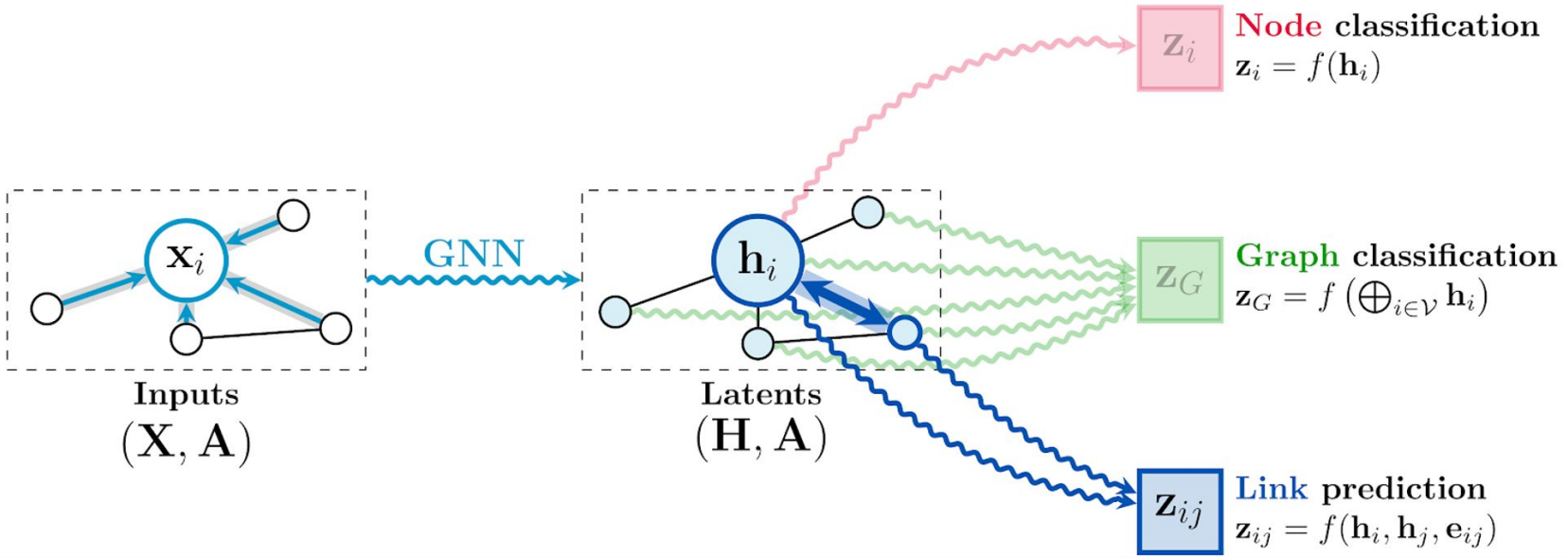
Node-level labels
- e.g. detect topics of documents based on citation graph

Edge-level labels (or edge existence; “link prediction”)
- e.g. detect side-effects between pairs of drugs in a drug-drug interaction graph

Graph-level labels
- e.g. predict chemical properties of a compound based on its atoms and bonds

When using neural networks: use appropriate loss function (e.g. cross-entropy for classification or mean-squared error for regression) and optimise by gradient descent + backpropagation.
Existing libraries (e.g. PyG, DGL, TF-GNN, Jraph) support this functionality out-of-the-box.

Tasks on a graph



Graph Neural Network Architectures

We use functions ψ and ϕ as neural networks operating over flat vector inputs.

The simplest example is a fully-connected MLP layer, e.g.:

$$\psi(\mathbf{x})=\sigma(\mathbf{W}\mathbf{x}+\mathbf{b}) \qquad \psi(\mathbf{x}, \mathbf{y})=\sigma(\mathbf{W}\mathbf{x}+\mathbf{U}\mathbf{y}+\mathbf{b})$$

$\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ are weights and biases, and σ is an activation function
(stacking more layers, e.g. $\psi(\mathbf{x}, \mathbf{y})=\sigma_2(\mathbf{W}_2\sigma_1(\mathbf{W}_1\mathbf{x}+\mathbf{U}\mathbf{y}+\mathbf{b}_1)+\mathbf{b}_2)$ is possible, and occasionally necessary)

The parameters are usually trainable via stochastic gradient descent.

Convolutional GNN

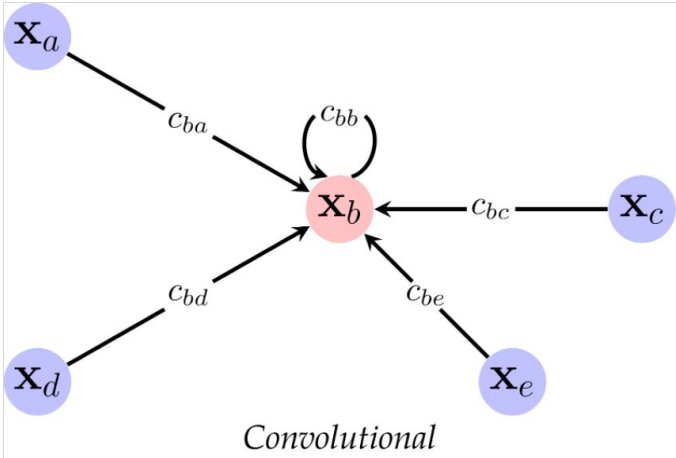
Features of neighbours aggregated with fixed weights, c_{ij}

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right)$$

Usually, weights depend directly on $\mathbf{A}$

- ChebyNet (Defferrard et al., NeurIPS'16)
- GCN (Kipf & Welling, ICLR'17)
- SGC (Wu et al., ICML'19)

Useful for homophilous graphs (when edges encode label similarity)
Highly scalable
Most industrial GNN applications currently live here



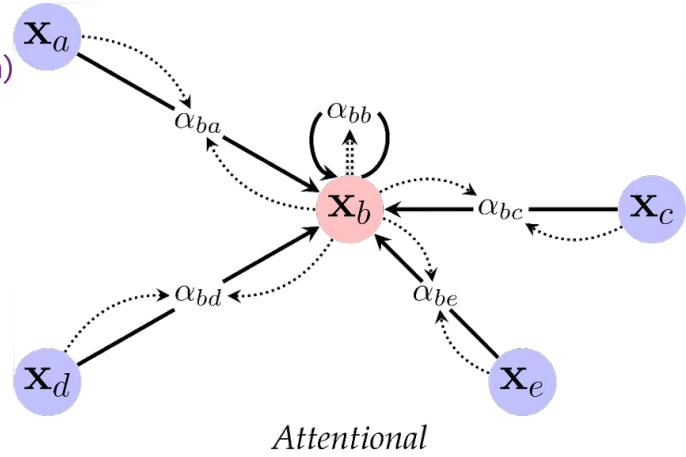
Convolutional

Mila

Attentional GNN

Features of neighbours aggregated with implicit weights (attention)

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right)$$



Attention weights computed as $\alpha_{ij}=a(\mathbf{x}_i,\mathbf{x}_j)$

- MoNet (Monti et al., CVPR'17)
- GAT (Veličković et al., ICLR'18)
- GATv2 (Brody et al., ICLR'22)

Useful as "middle ground" w.r.t. capacity, scale, interpretability
Edges need not encode homophily
But still computing only a scalar per edge
Can "imitate" CNNs (local, efficient, anisotropic)

Message-passing GNN

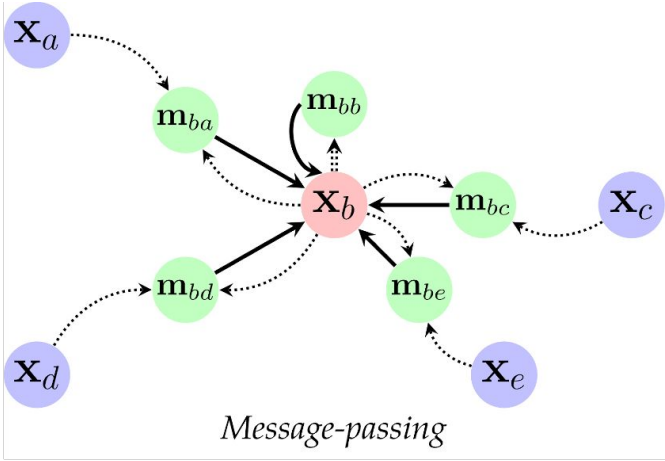
Compute arbitrary vectors (messages) to be sent across edges

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

Messages computed as $\mathbf{m}_{ij}=\psi(\mathbf{x}_i,\mathbf{x}_j)$

- Interaction Nets (Battaglia et al., NeurIPS'16)
- MPNN (Gilmer et al., ICML'17)
- GraphNets (Battaglia et al., 2018)

Most generic GNN layer
Edges give "recipe" for passing data
May have scalability or learnability issues
Ideal for computational chemistry, reasoning and simulation tasks



Mila

General attributed graphs

Especially here, graphs can convey rich information at all granularities

We can assume a more general possible set of features:

- Node features, $\mathbf{x}_u \in \mathbb{R}^k$
- Edge features, $\mathbf{x}_{uv} \in \mathbb{R}^l$
- Graph features, $\mathbf{x}_{\mathcal{G}} \in \mathbb{R}^m$

Analogously defining latents $\mathbf{h}_u$, $\mathbf{h}_{uv}$, $\mathbf{h}_{\mathcal{G}}$

It is possible to extend this further (e.g. hypergraphs)
but such inputs can usually be represented as an instance of the above

Graph Networks

Update edge features (using graph + relevant nodes)

$$\mathbf{h}_{uv} = \psi\left(\mathbf{x}_u, \mathbf{x}_v, \mathbf{x}_{uv}, \mathbf{x}_{\mathcal{G}}\right)$$

Update node features (using updated relevant edges + graph)

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{u \in \mathcal{N}_v} \mathbf{h}_{vu}, \mathbf{x}_{\mathcal{G}}\right)$$

Update graph features (using updated nodes + edges)

$$\mathbf{h}_{\mathcal{G}} = \rho\left(\bigoplus_{u \in \mathcal{V}} \mathbf{h}_u, \bigoplus_{(u,v) \in \mathcal{E}} \mathbf{h}_{uv}, \mathbf{x}_{\mathcal{G}}\right)$$

Part II: Graph (re)wiring

Where does the graph come from?

So far, we've assumed that the "ground-truth" graph is given to us!

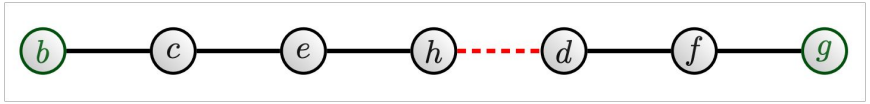
The vertices are the entities we care about.

The adjacency:

- k-nearest neighbours for each vertex
- vertices that are within a distance range
- bonds in molecules, train connections in transportation networks, co-authorship in citation networks...

Where does the graph come from?

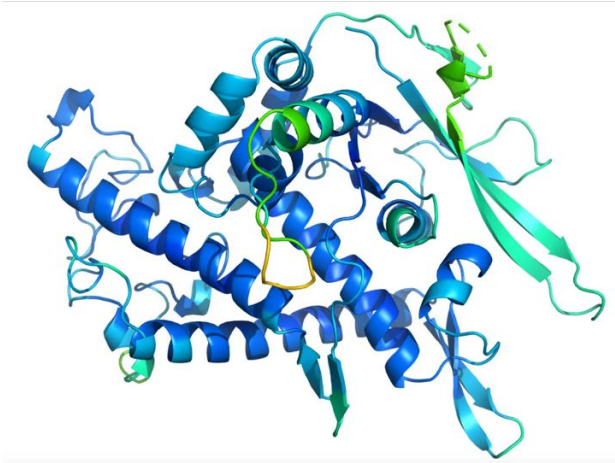
Even when graph is perfectly correct, it might not be optimal for the task
For path-querying, many graphs “fully correctly” encode the task
One induces $O(n)$ GNN layers; the other induces near-$O(1)$ layers!



Where does the graph come from?

Even when graph is perfectly correct, it might not be optimal for the task
For path-querying, many graphs “fully correctly” encode the task
One induces $O(n)$ GNN layers; the other induces near-$O(1)$ layers!

In the real world, “perfect graphs” are incredibly rare
Even for “standard” inputs like molecules…
All atom pairs interact, not just across bonds!

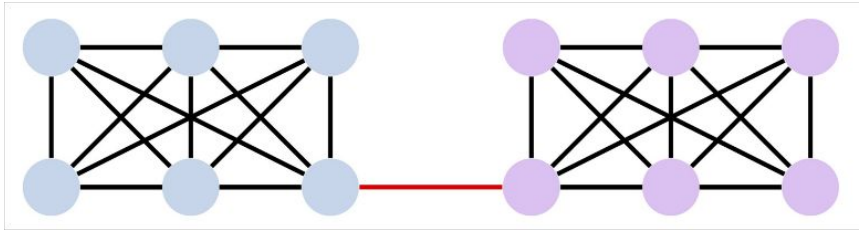


Where does the graph come from?

Even when graph is perfectly correct, it might not be optimal for the task
For path-querying, many graphs “fully correctly” encode the task
One induces $O(n)$ GNN layers; the other induces near-$O(1)$ layers!

In the real world, “perfect graphs” are incredibly rare
Even for “standard” inputs like molecules…
All atom pairs interact, not just across bonds!

Functions on some graphs might be particularly hard to model, pushing us in new territory when having bottlenecks, long-range interactions, heterophily…

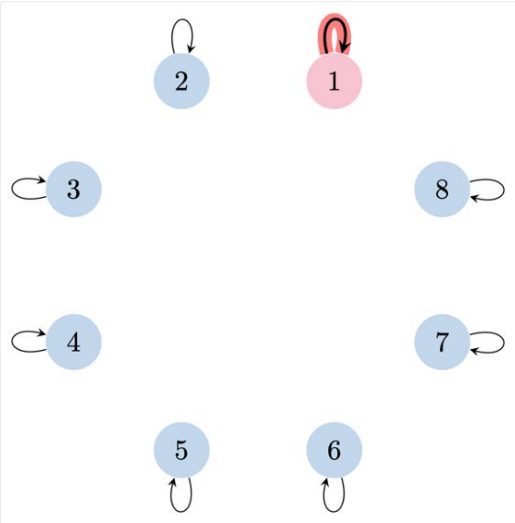


Solution 1: ~no edges

"Pessimistic" approach: assume the graph is completely disconnected
Let $\mathbf{A}=\mathbf{I}$, or, equivalently, $\mathcal{N}_u=\{u\}$ for all nodes u

Any GNN flavour reduces to Deep Sets:

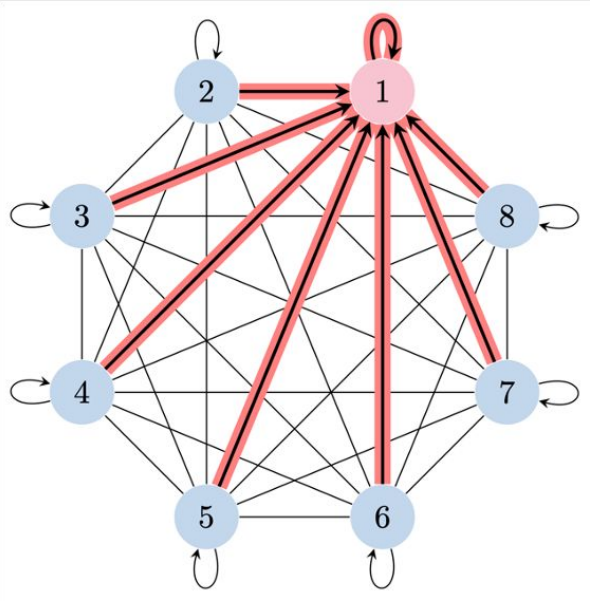
$$\mathbf{h}_u = \psi(\mathbf{x}_u)$$



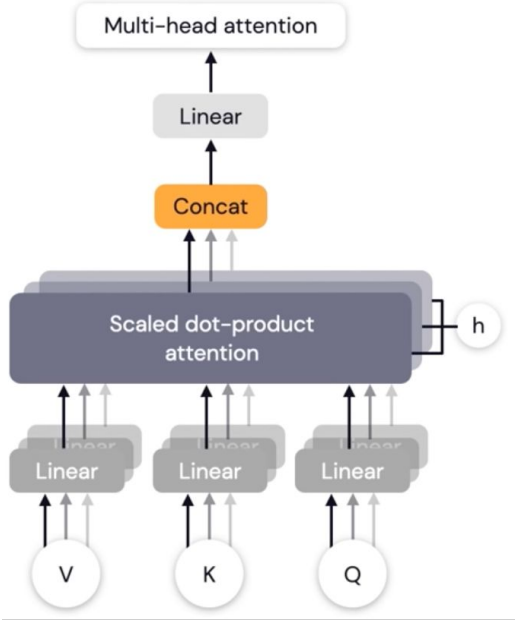
Mila

Solution 2: all edges

"Lazy" approach: assume the graph is fully connected
Let $\mathbf{A}=\mathbf{1}\mathbf{1}^{\top}$, or, equivalently, $\mathcal{N}_u=\mathcal{V}$ for all nodes u



Solution 2: all edges



"Lazy" approach: assume the graph is fully connected
Let $\mathbf{A}=\mathbf{1}\mathbf{1}^T$, or, equivalently, $\mathcal{N}_u=\mathcal{V}$ for all nodes u

If we use attentional GNNs:

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \sum_{v\in\mathcal{V}} a(\mathbf{x}_u, \mathbf{x}_v)\psi(\mathbf{x}_v)\right)$$

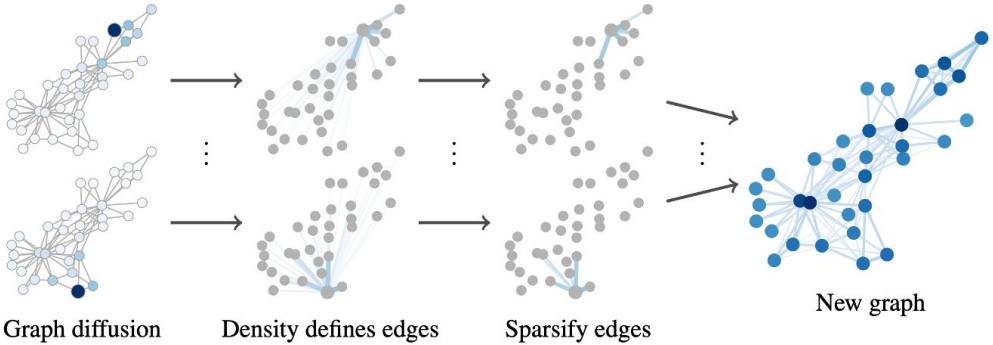
we recover Transformers.

The sequential structural information is injected through the positional embeddings, but the model is under no requirement to use this information.
Attention ~ inferring soft adjacency

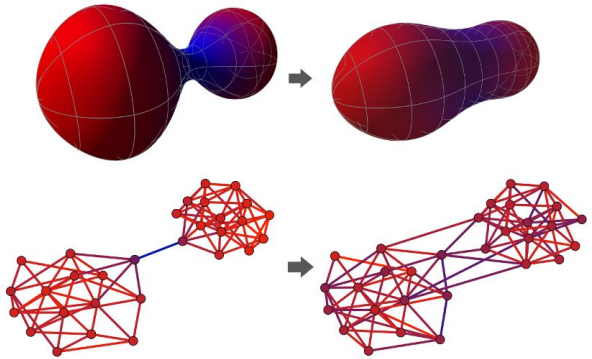
Mila

Solution 3: Nonparametric rewiring

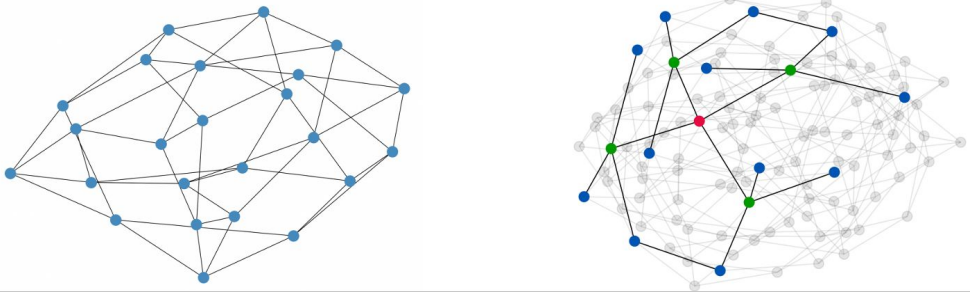
Graph diffusion convolution
(GDC; Gasteiger et al., NeurIPS'19)



Stochastic Discrete Ricci Flow
(SDRF, Topping, di Giovanni et al, ICLR'22)

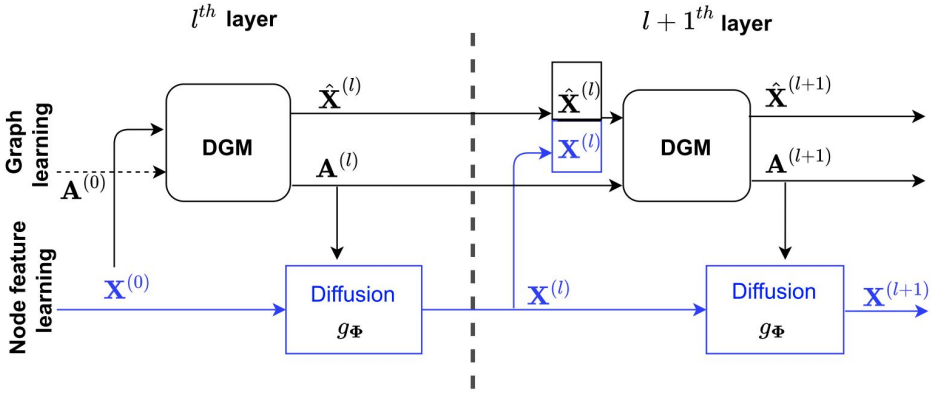


Expander Graph Propagation (EGP, Deac et al, LoG'22)

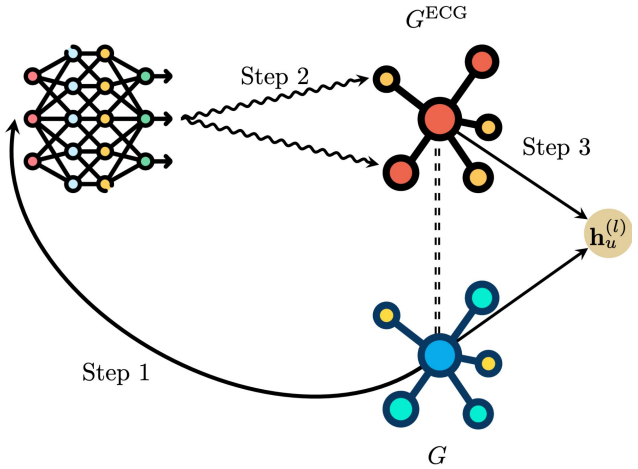


Solution 4: Parametric rewiring

Differentiable Graph Module
(DGM, Kazi et al '23)



Evolving Computation Graphs
(ECG, Deac and Tang '23)

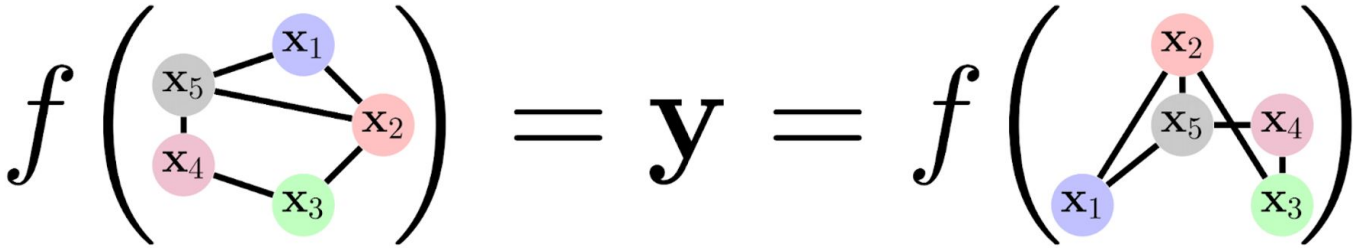


Mila

Part III: Geometric Deep Learning

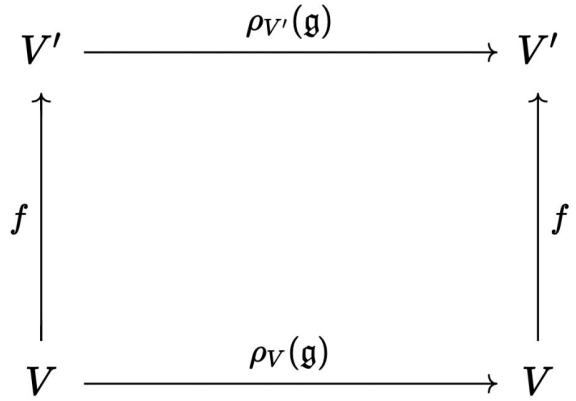
Through the symmetry lens

GNNs have a key property: permuting the nodes and edges transforms the results predictably.



Equivariance

We say a function, f : V → V', is $\mathfrak{G}$-equivariant (for a symmetry group $\mathfrak{G}$), if for all $\mathfrak{g} \in \mathfrak{G}$:



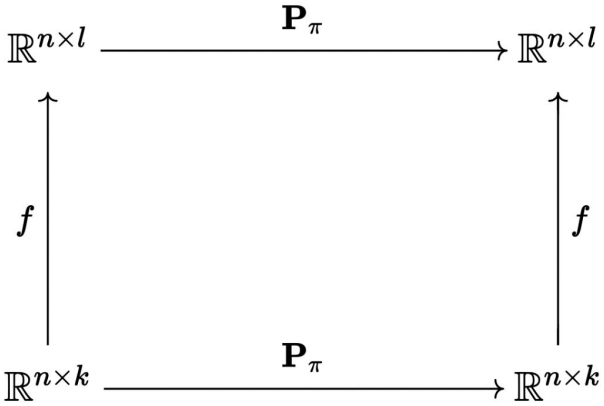
$$f \circ \rho_V(\mathfrak{g}) = \rho_{V'}(\mathfrak{g}) \circ f$$

Forcing neural networks to respect $\mathfrak{G}$-equivariance means a much smaller hypothesis space!
⇒ Higher data efficiency, better OOD generalisation, safer (more consistent), etc.

Permutation equivariance

We can, for example, look at node representations over the permutation group.

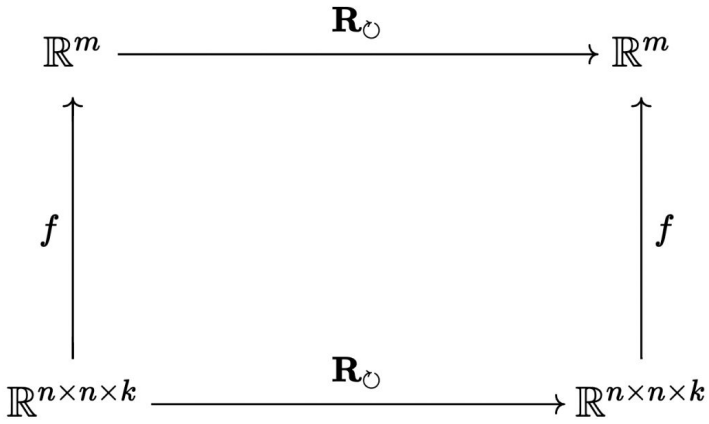
This lets us recover models like Deep Sets, Graph Neural Networks, and Transformers!



$$f(\mathbf{PX}) = \mathbf{P}f(\mathbf{X})$$

Rotation equivariance

To enforce SO(2) equivariance, we adapt our diagram as before.
Now, the state embedding is equivariant to 2D rotations. A rotation of the input grid implies a rotation of the representations!



$$f(\mathbf{R}_{\circlearrowleft}\mathbf{X}) = \mathbf{R}_{\circlearrowleft} f(\mathbf{X})$$

Achieving equivariance in NNs

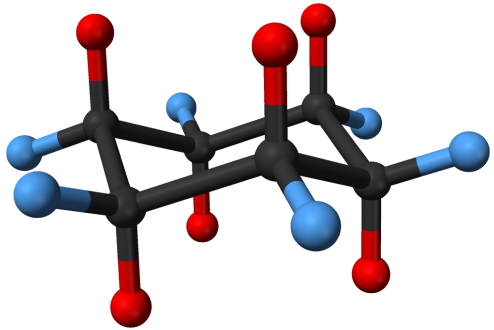
Property	Train augmentation	Test augmentation	Equivariance
Layerwise or whole-net constraint	Whole-net	Whole-net	Layerwise
Easy to implement, simple			
Guaranteed in/equivariance at training data			
Guaranteed in/equivariance at test data			
Works for large (e.g S_n) / infinite (e.g. SE(n)) groups			
Efficient at train time			
Efficient at test time			

Geometric graphs

Thus far, we have assumed our graphs to be a discrete, unordered, set of nodes and edges.

The graph, in fact, may often be endowed with some spatial geometry.
It will be useful for us to exploit this geometry!

Molecules are a classical case (with their three-dimensional conformers)



Learning over geometric graphs

Simplified setup: nodes endowed with features, f_u and coordinates $x_u \in \mathbb{R}^3$

An equivariant message passing layer transforms them separately
Yielding updated features f'_u and coordinates x'_u

We can now express a group of symmetries $\mathfrak{G}$ we would like to be resistant to
In the case of molecules, a standard group is the Euclidean group, E(3) of roto-translations and reflections.

For any 3D orthogonal matrix R and translation vector b, we define a group action $\mathfrak{g} \in E(3)$:

$$\rho(\mathfrak{g})\mathbf{x} = \mathbf{R}\mathbf{x} + \mathbf{b}$$

and applying them to coordinates typically should not affect how features are processed!

E(n)GNN

There exist many GNNs that obey E(n)-equivariance.

One elegant solution was exposed by Satorras et al. (ICML'21):

$$\mathbf{f}'_u = \phi\left(\mathbf{f}_u, \bigoplus_{v \in \mathcal{N}_u} \psi_{\mathrm{f}}(\mathbf{f}_u, \mathbf{f}_v, \|\mathbf{x}_u - \mathbf{x}_v\|^2)\right),$$

$$\mathbf{x}'_u = \mathbf{x}_u + \sum_{v \neq u} (\mathbf{x}_u - \mathbf{x}_v)\psi_{\mathrm{c}}(\mathbf{f}_u, \mathbf{f}_v, \|\mathbf{x}_u - \mathbf{x}_v\|^2)$$

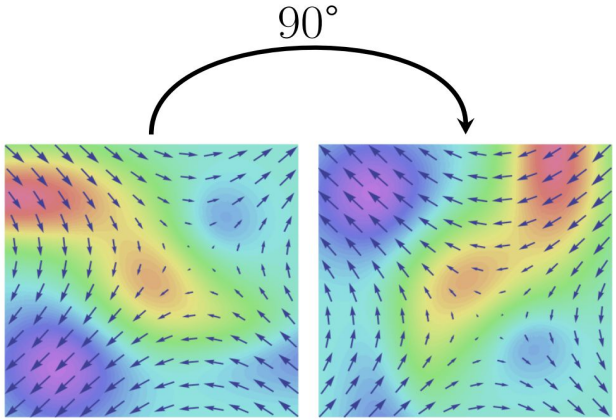
The actions of E(n) do not change distance between nodes (they are isometric)
Hence if $x_u \leftarrow Rx_u + b$...
f'_u does not change, and $x'_u \leftarrow Rx'_u + b$, as expected!

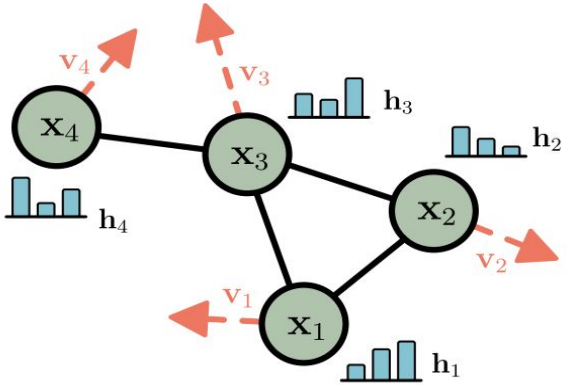
Hence, this layer is E(n)-equivariant over scalar features f_u

Vector-structured features

However, what if some of the node features (f_u) depend on the geometry?
e.g. they could be vector forces
Rotating the molecule should rotate these vectors too!

The model on the previous slide does not take this into account!





Mila

Vector-structured features

However, what if some of the node features (f_u) depend on the geometry?
e.g. they could be vector forces
Rotating the molecule should rotate these vectors too!

The model on the previous slide does not take this into account!

Satorras et al. do propose a variant of their model that works with vectors:

$$\mathbf{v}_i^{l+1} = \phi_v\left(\mathbf{h}_i^l\right)\mathbf{v}_i^l + C\sum_{j\neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{ij}\right)$$

$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + \mathbf{v}_i^{l+1}$$

But the issue will keep reappearing as we “tensor up” our features!
Is there a way to talk about the general “set of solutions” for E(n)-equivariance?

Towards a general solution: Tensor Field Networks

We actually can factorise the group action of SE(3) on inputs of any tensor order l

$$\rho(g) = \mathbf{Q}^\top \left[\bigoplus_\ell \mathbf{D}_\ell(g) \right] \mathbf{Q}$$

where D_l are the Wigner D-matrices, and Q is a $\mathfrak{g}$-specific change-of-basis matrix.

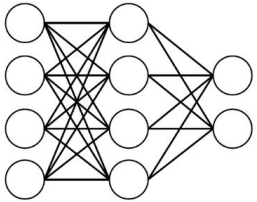
This can be exploited to write a generic SE(3)-equivariant model (the TFN)

$$\mathbf{f}_{\text{out},i}^\ell = \sum_{k \geq 0} \underbrace{\int \mathbf{W}^{\ell k}(\mathbf{x}' - \mathbf{x}_i) \mathbf{f}_{\text{in}}^k(\mathbf{x}') \, \mathrm{d}\mathbf{x}'}_{k \to \ell \text{ convolution}} = \sum_{k \geq 0} \sum_{j=1}^{n} \underbrace{\mathbf{W}^{\ell k}(\mathbf{x}_j - \mathbf{x}_i) \mathbf{f}_{\text{in},j}^k}_{\text{node } j \to \text{node } i \text{ message}},$$

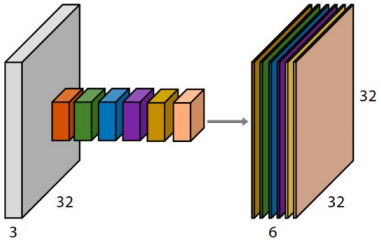
$$\mathbf{W}^{\ell k}(\mathbf{x}) = \sum_{J=|k-\ell|}^{k+\ell} \varphi_J^{\ell k}(\|\mathbf{x}\|) \mathbf{W}_J^{\ell k}(\mathbf{x}), \qquad \text{where } \mathbf{W}_J^{\ell k}(\mathbf{x}) = \sum_{m=-J}^{J} Y_{Jm}(\mathbf{x}/\|\mathbf{x}\|) \mathbf{Q}_{Jm}^{\ell k}$$

where the matrices W_J are precomputed based on the above parametrisation.

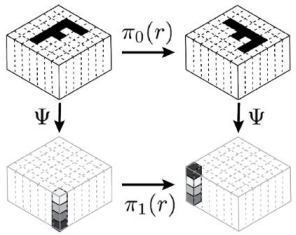
Geometric Deep Learning



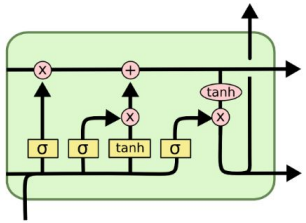
Perceptrons
Function regularity



CNNs
Translation

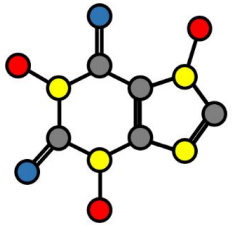


Group-CNNs
Translation+Rotation,
Global groups



LSTMs
Time warping

DeepSets / Transformers
Permutation



GNNs
Permutation

Intrinsic CNNs
Isometry / Gauge choice

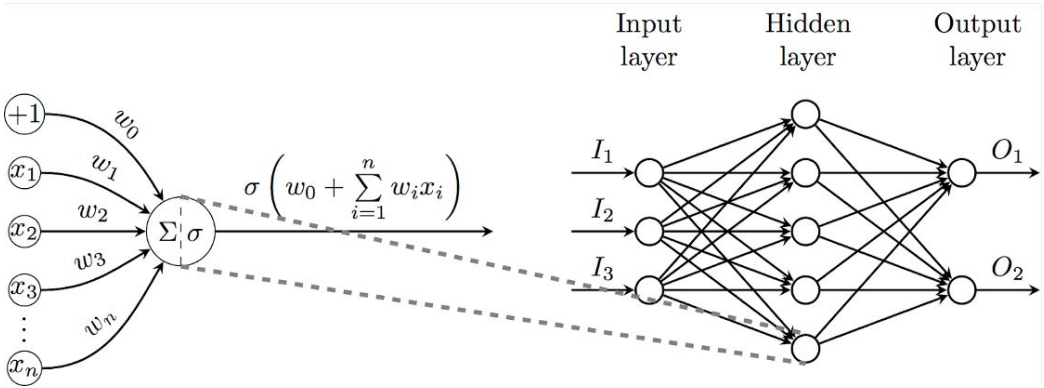
Part IV: Simulation/Algorithmic reasoning

MLP

Input: (flat) representation of an object's features
(e.g. position, shape, color...)

Output: some property of the object
(e.g. is it round and yellow?)

A canonical problem solvable by a multilayer perceptron (MLP).

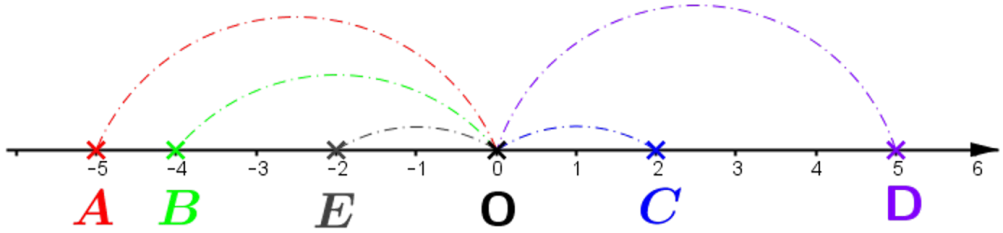


Mila

Deep Sets

Input: A set of 1D points, with features containing their coordinate and colour.

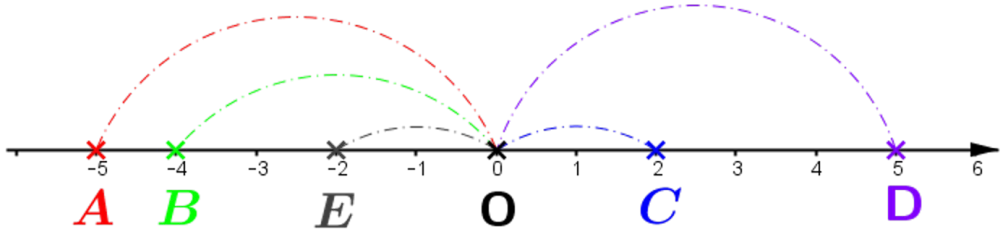
Output: Some aggregate property of the set
(e.g. the furthest pairwise distance).



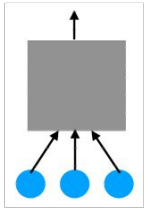
GNNs

Input: A set of 1D points, with features containing their coordinate and colour.

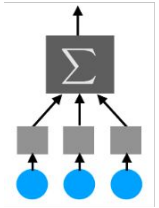
Output: Some relational property of the set
(e.g. the colours of the two furthest points).



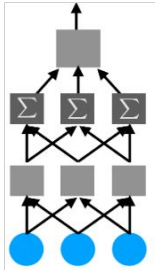
Architecture ⟷ Task



MLP: feature extraction



Deep Sets: summary statistics



GNN: (pairwise) relations

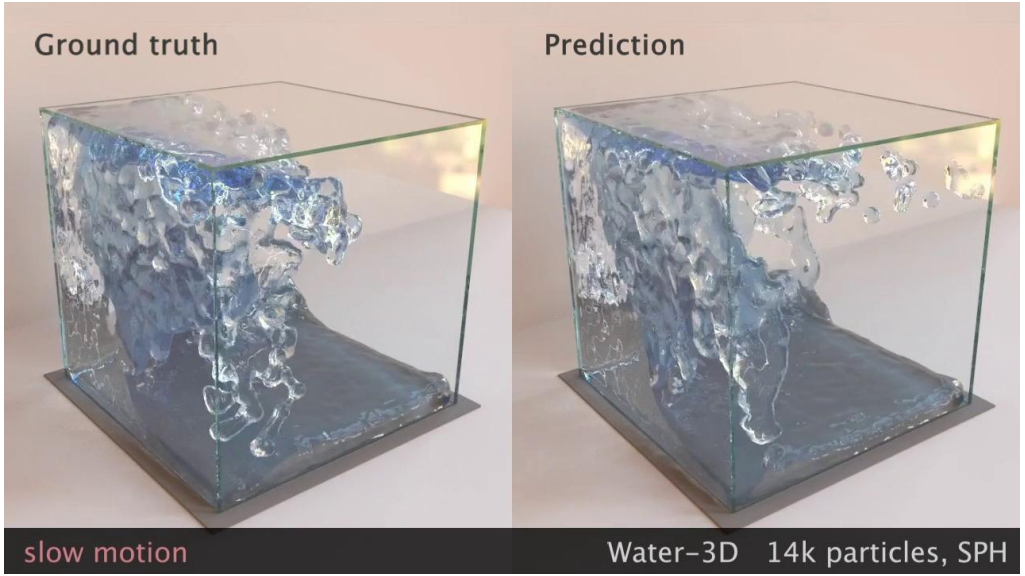
Not all architectures are equal for a task!
Becomes obvious in terms of sample complexity.

“What can neural networks reason about?”
(Xu et al., ICLR’20)

Theorem:
Better structural alignment
leads to better generalisation!

Physical simulations

Learning to simulate complex physics with graph networks, Sanchez-Gonzalez et al, ICML 2020
Graph of particles, predict delta in position for each particle for each time step.



Mila

Simulations

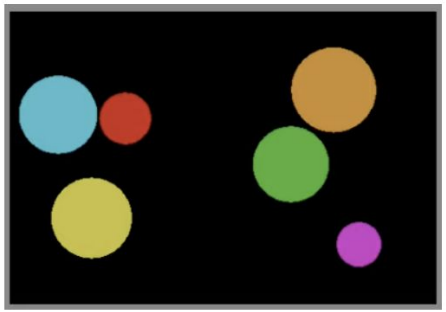
Some takeaways:

- factorized, knowledgeable representation makes the problem more tractable
- use the GNN processor for one-step of forward dynamics
- predict delta
- recall (feed in t_0 input at different timesteps)
- causal regularization can be useful (e.g. from systems that have the same trajectories)
- ...

What happens when the input is not processed?

Physical simulations

Natural inputs



$\mathbf{x}$

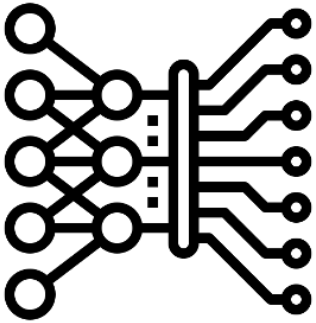
????

Natural outputs

$\mathbf{y}$

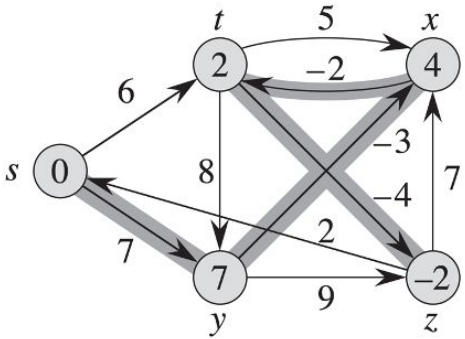
NNs vs Algorithms

Neural Networks



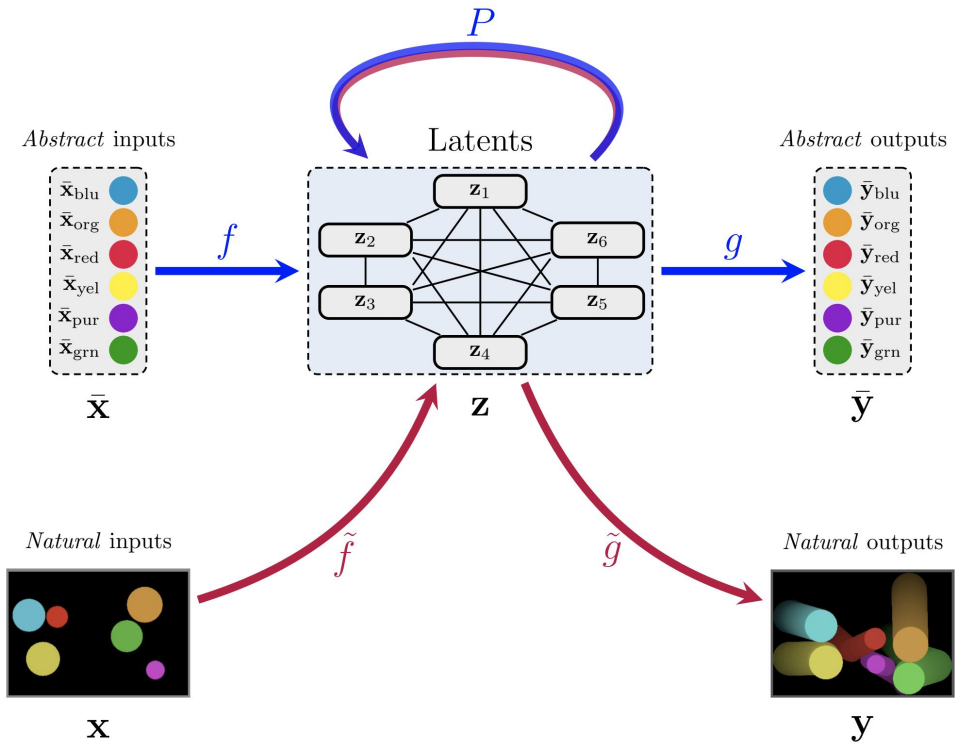
+ Operate on raw inputs
+ Generalise on noisy conditions
+ Models reusable across tasks
- Require big data
- Unreliable when extrapolating
- Lack of interpretability

Algorithms



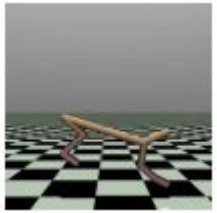
+ Trivially strongly generalise
+ Compositional (subroutines)
+ Guaranteed correctness
+ Interpretable operations
- Inputs must match spec
- Not robust to task variations

Neural Algorithmic Reasoning



Neural Algorithmic Reasoning

- Previous slide: Reasoning Modulated Representations, Velickovic et al, LoG 2022
- brain vessel classification
  - DUAL ALGORITHMIC REASONING, Numeroso et al, ICLR 2023
- planning in reinforcement learning:
  - Neural Algorithmic Reasoners are Implicit Planners, Deac et al, NeurIPS 2021
  - Continuous Neural Algorithmic Planners, Yu et al, LoG 2022


References

The Geometric Deep Learning book by Michael M. Bronstein, Joan Bruna, Taco Cohen, Petar Veličković – https://geometricdeeplearning.com/

Material from the course taught by Petar Veličković and Pietro Lio at University of Cambridge: Representation Learning on Graphs and Networks – https://www.cl.cam.ac.uk/teaching/2122/L45/

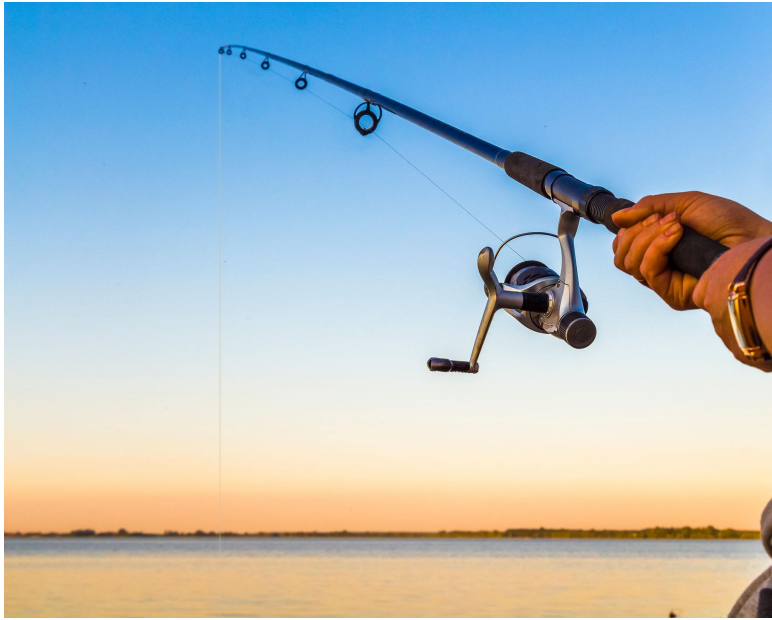
Tutorial at Learning on Graphs 2022, with Petar Veličković and Andrew Dudzik– https://algo-reasoning.github.io/

GNN Colabs

https://github.com/gordicaleksa/pytorch-gat

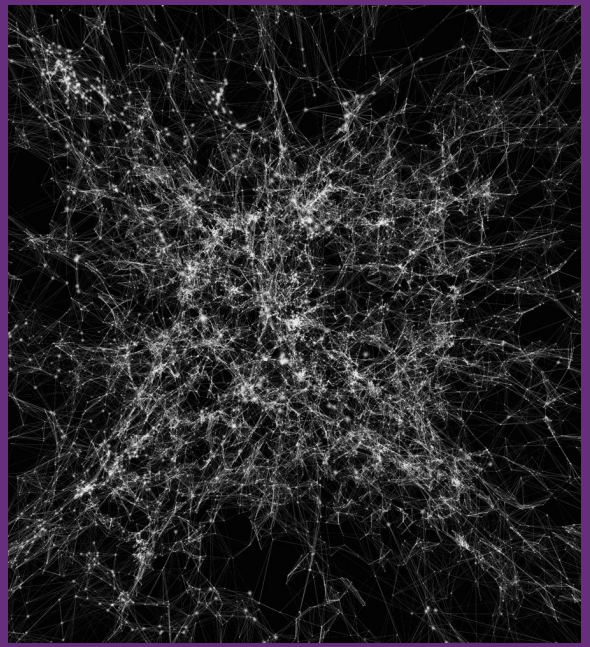
https://github.com/eemlcommunity/PracticalSessions2021/blob/main/graphnets/graphnets_tutorial.ipynb

https://github.com/deepmind/educational/blob/master/colabs/summer_schools/intro_to_graph_nets_tutorial_with_jraph.ipynb



Mila

Thank you!



andreeadeac22{@gmail.com, @X, @LinkedIn}