Designing and explaining graph neural networks

Presenter: Simone Scardapane

SAPIENZA
UNIVERSITÀ DI ROMA

intelligent signal processing
and multimedia lab

Baker Hughes seminar, 2023

Not just me!

Simone Scardapane
Tenure-track Assistant Professor

https://www.sscardapane.it/

https://twitter.com/s_scardapane

Aurelio Uncini
Full professor

Massimo Panella
Full professor

Indro Spinelli
Researcher (comp. science)

Paolo Di Lorenzo
Associate professor

Filippo Bianchi
Associate professor (UiT)

Alessio Verdone
PhD

Lev Telyatnikov
PhD

Alessio Devoto
PhD

Alessandro Baiocchi
PhD

Gaetano Saurio
PhD

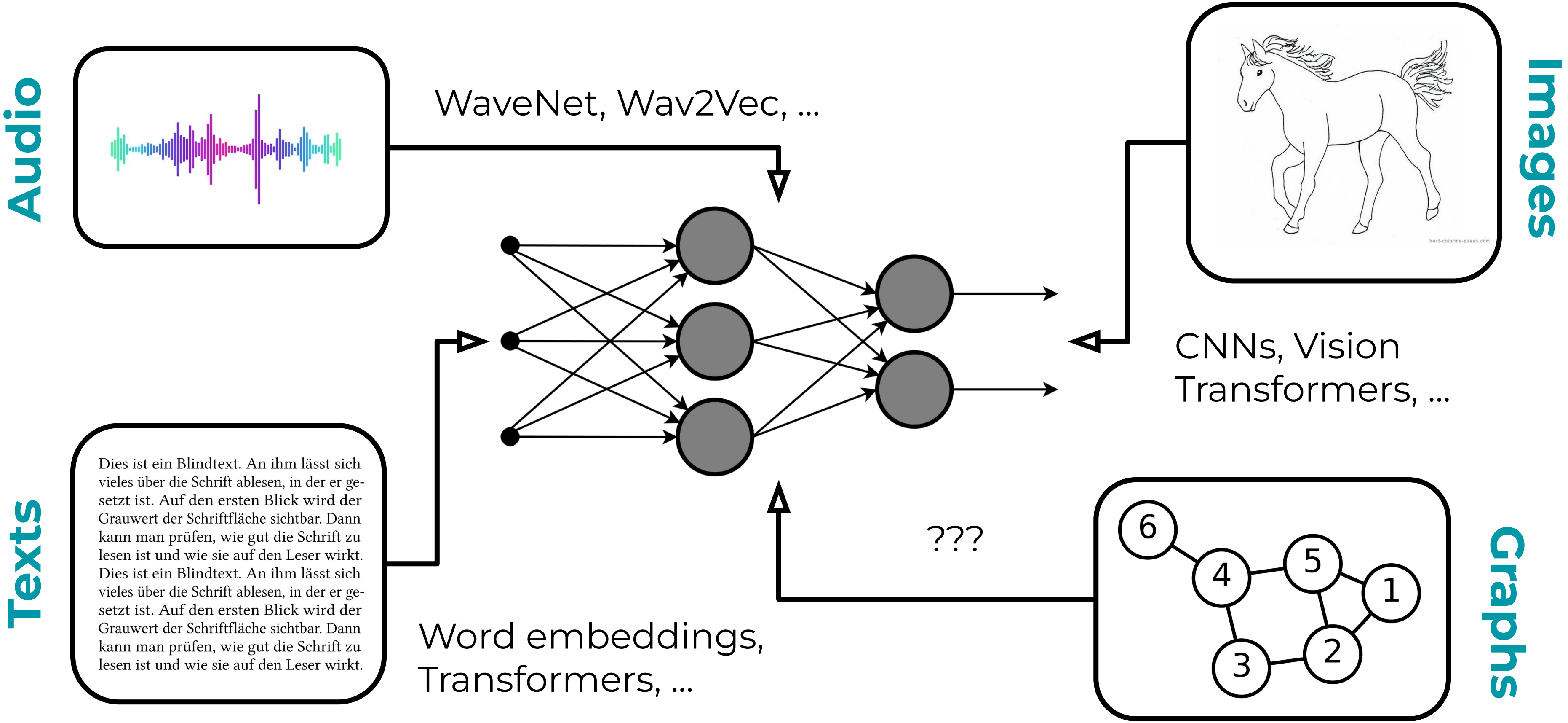
Michele Guerra
PhD (UiT)

Claudio Battiloro
PhD

Introduction

On the importance of **graphs** in **deep learning**

Data ingestion in deep learning



Audio

WaveNet, Wav2Vec, ...

Images

CNNs, Vision Transformers, ...

Texts

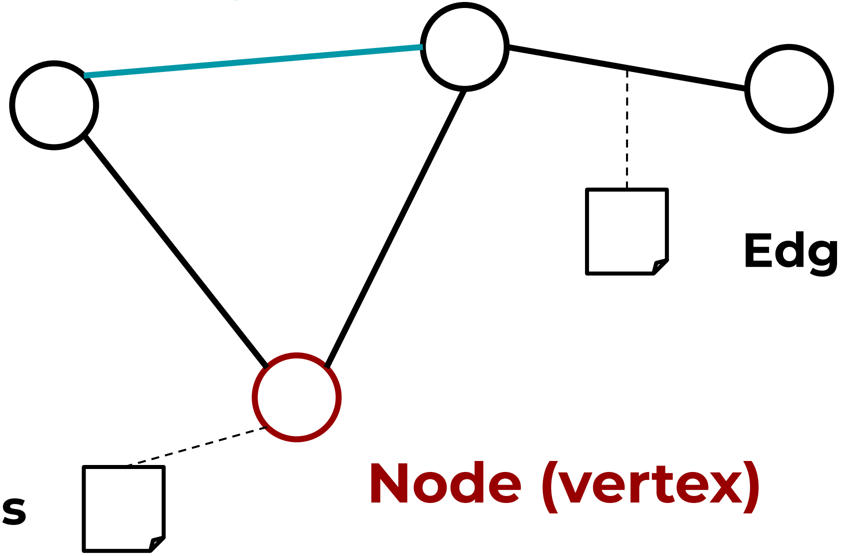
Dies ist ein Blindtext. An ihm lässt sich vieles über die Schrift ablesen, in der er gesetzt ist. Auf den ersten Blick wird der Grauwert der Schriftfläche sichtbar. Dann kann man prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt. Dies ist ein Blindtext. An ihm lässt sich vieles über die Schrift ablesen, in der er gesetzt ist. Auf den ersten Blick wird der Grauwert der Schriftfläche sichtbar. Dann kann man prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt.

Word embeddings, Transformers, ...

Graphs

???

Defining a graph



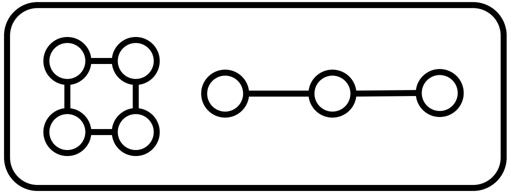
Edge

Edge features

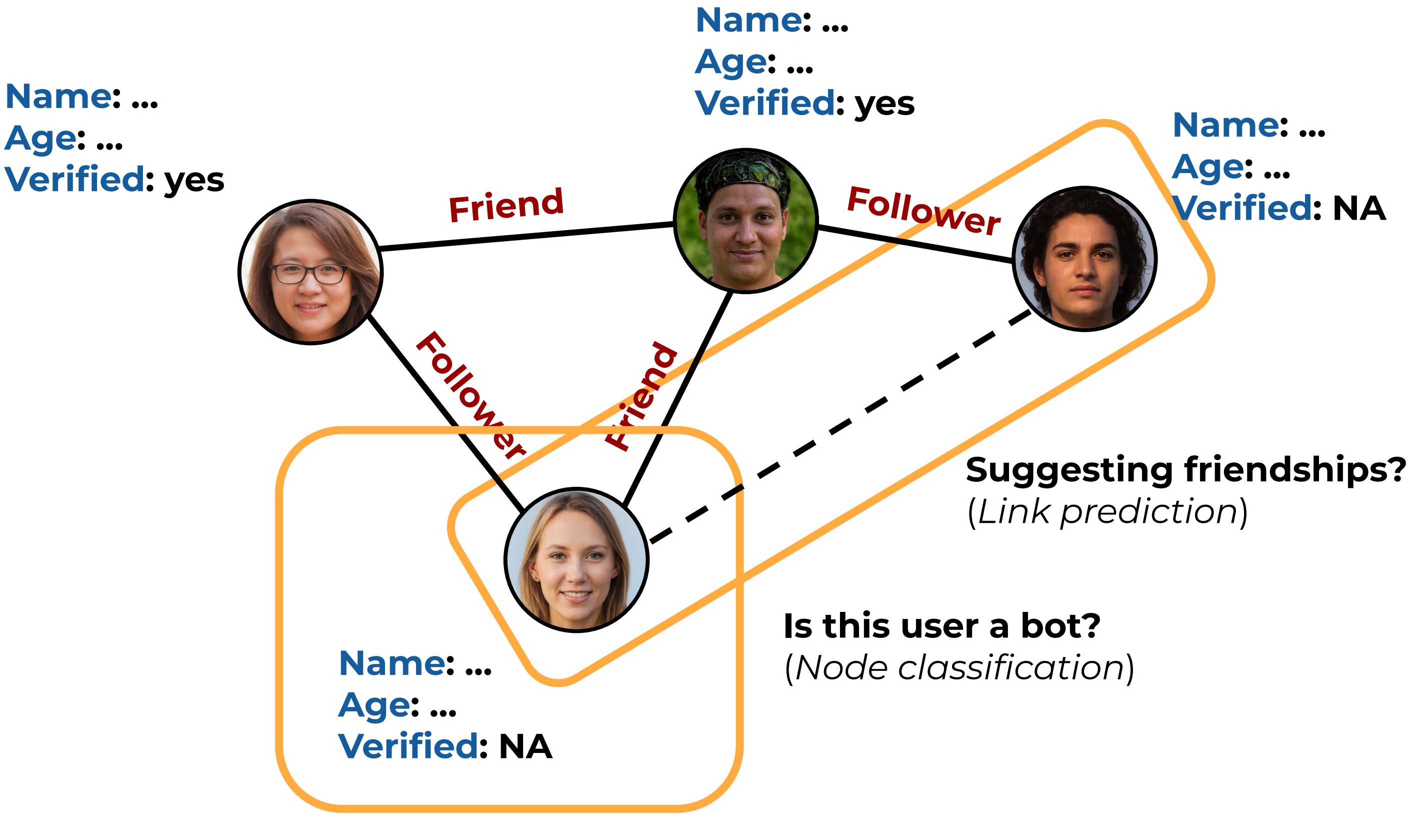
Node features

Node (vertex)

Other graphs!



Applications (1/3)



Suggesting friendships?
(Link prediction)

Is this user a bot?
(Node classification)

Fake news detection on Twitter

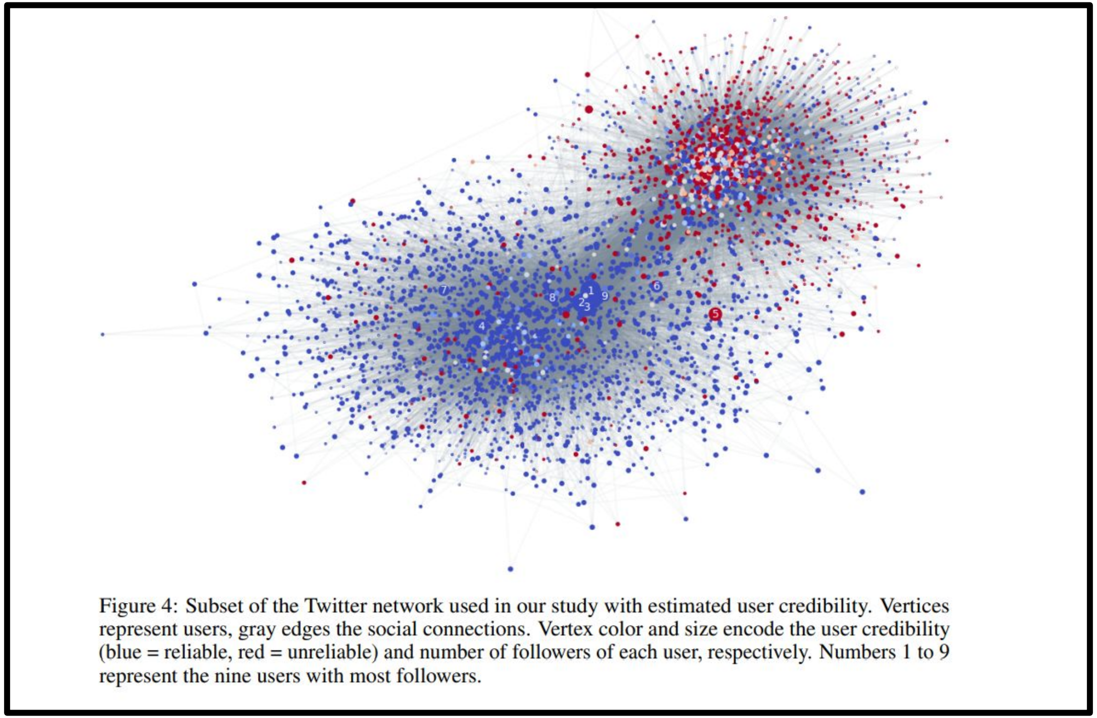
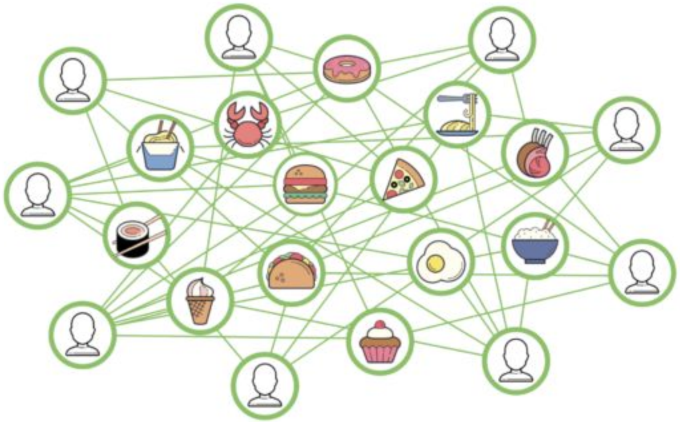
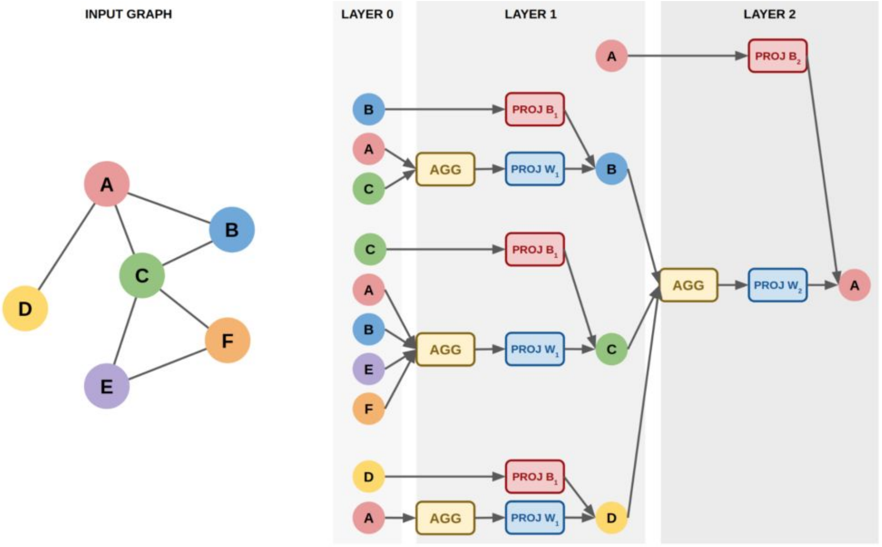


Figure 4: Subset of the Twitter network used in our study with estimated user credibility. Vertices represent users, gray edges the social connections. Vertex color and size encode the user credibility (blue = reliable, red = unreliable) and number of followers of each user, respectively. Numbers 1 to 9 represent the nine users with most followers.

Monti, F., Frasca, F., Eynard, D., Mannion, D. and Bronstein, M.M., 2019. **Fake News Detection on Social Media using Geometric Deep Learning**. *arXiv preprint arXiv:1902.06673.*

Recommending systems in Uber

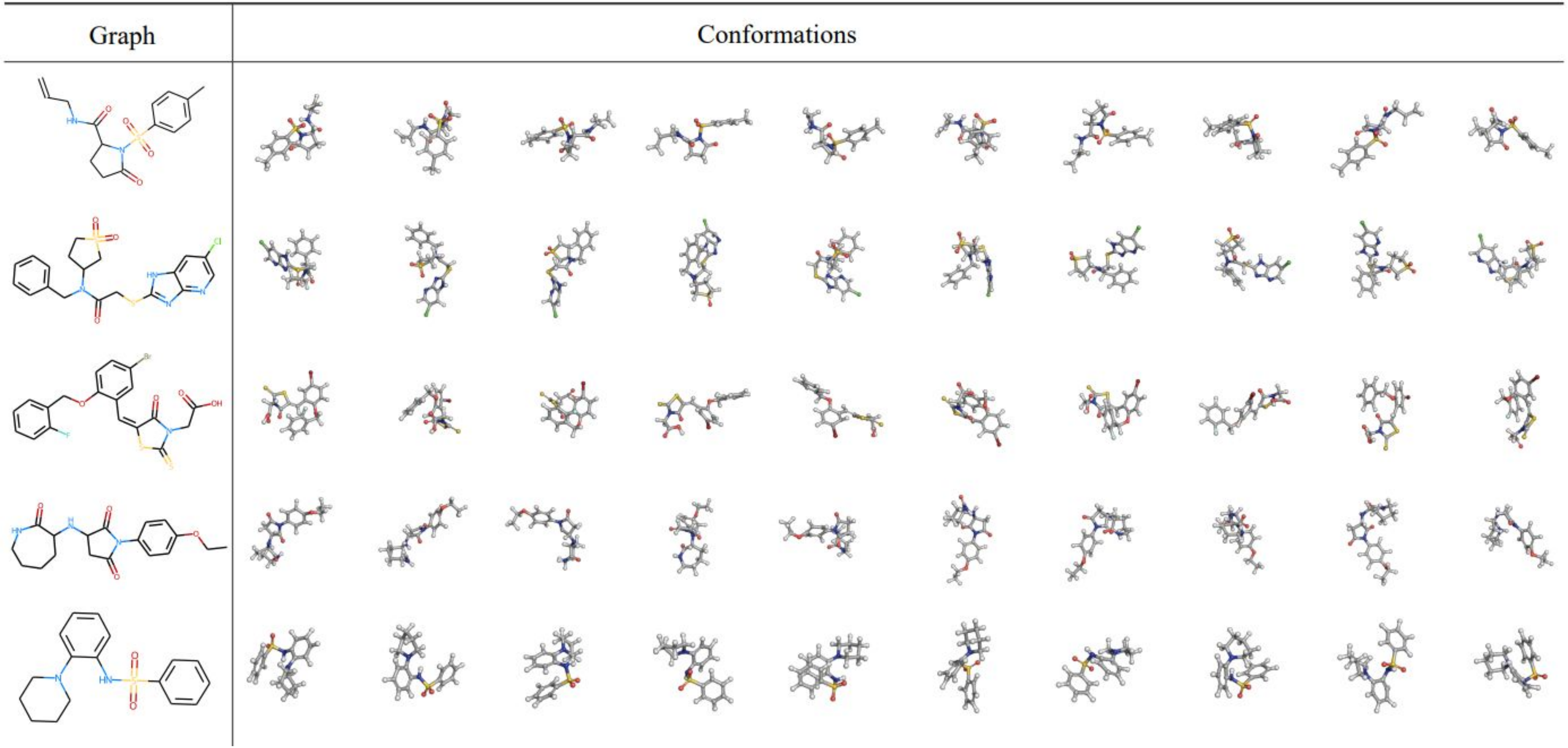




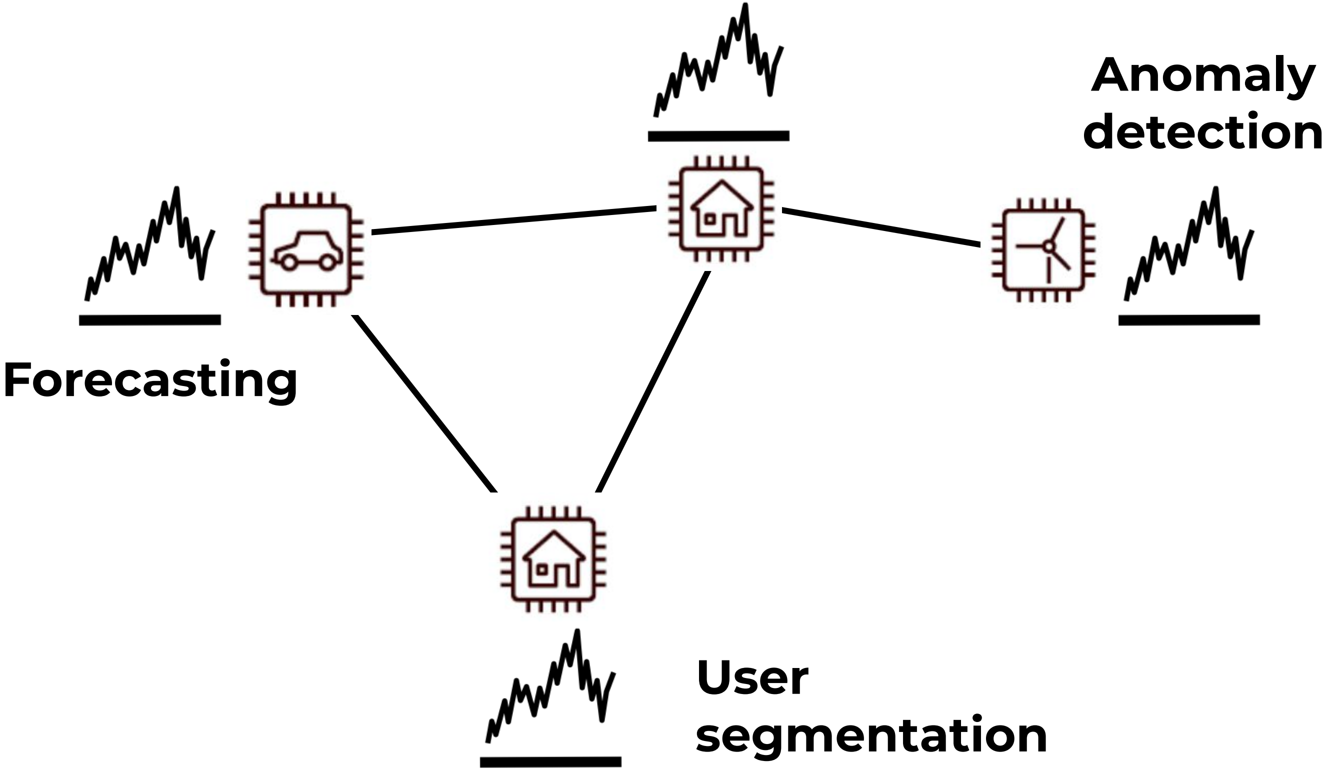
[Food Discovery with Uber Eats: Using Graph Learning to Power Recommendations]

Applications (2/3)

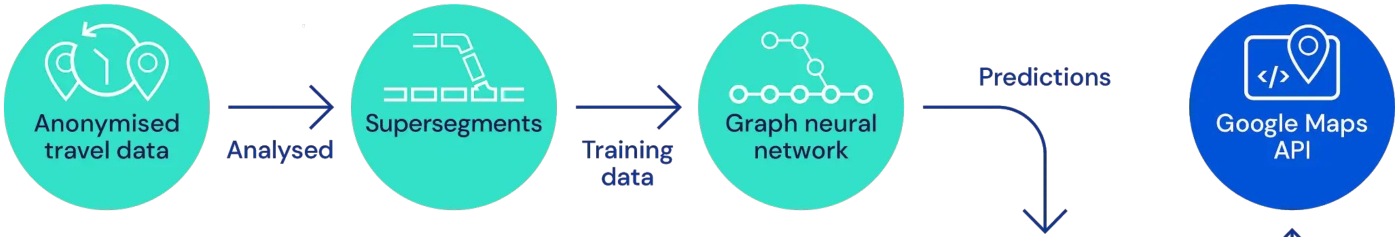
[2203.02923] GeoDiff: a Geometric Diffusion Model for Molecular Conformation Generation

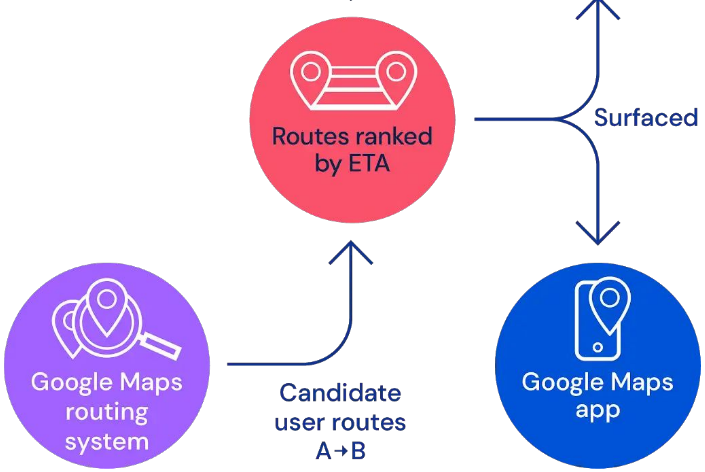


Applications (3/3)

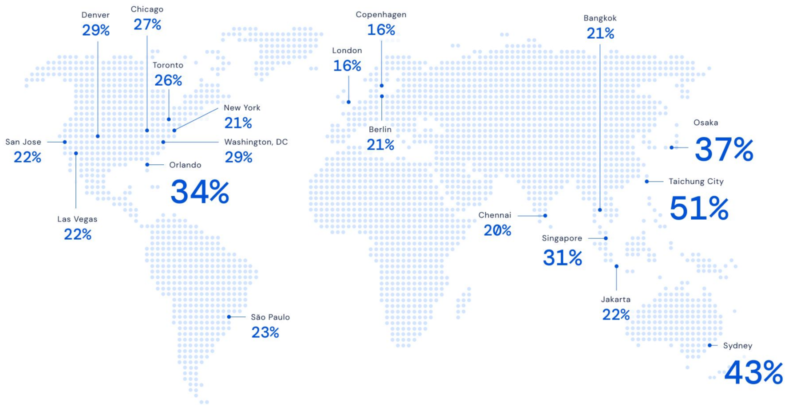


Traffic prediction on Google Maps



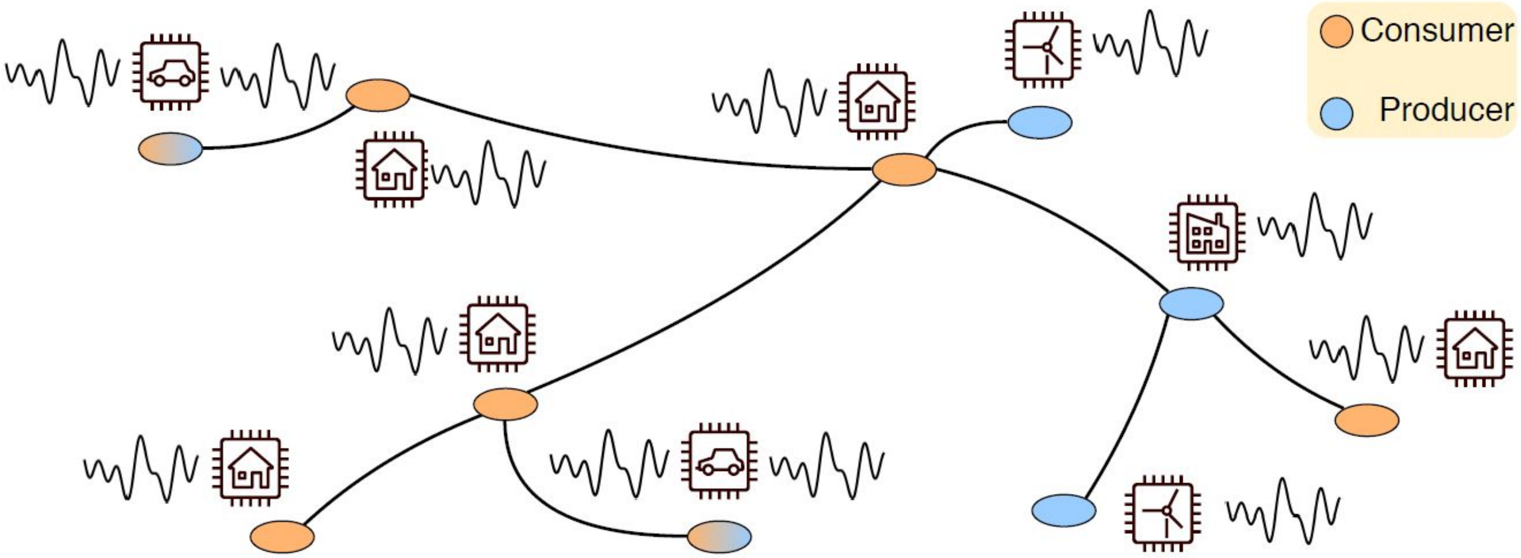


Google Maps ETA Improvements Around the World



Traffic prediction with advanced Graph Neural Networks

Distributed energy grids



What happens when our graph is also **physically distributed**?

Energy forecasting on smart-grids

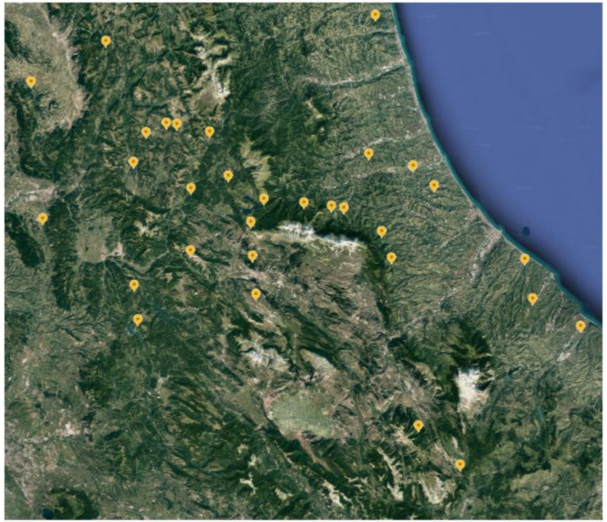


Fig. 2. Map showing the distribution of 31 simulated photovoltaic plants.

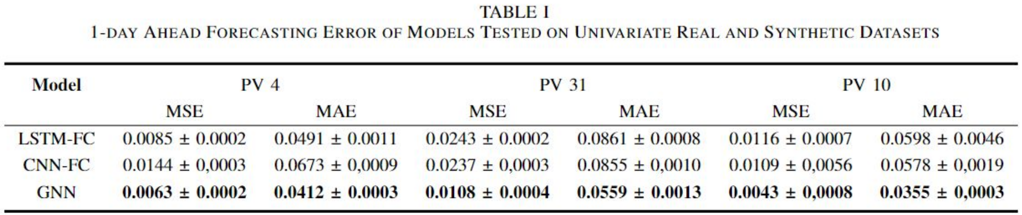
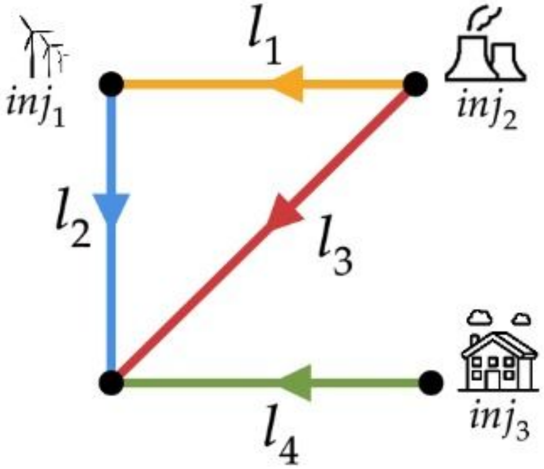


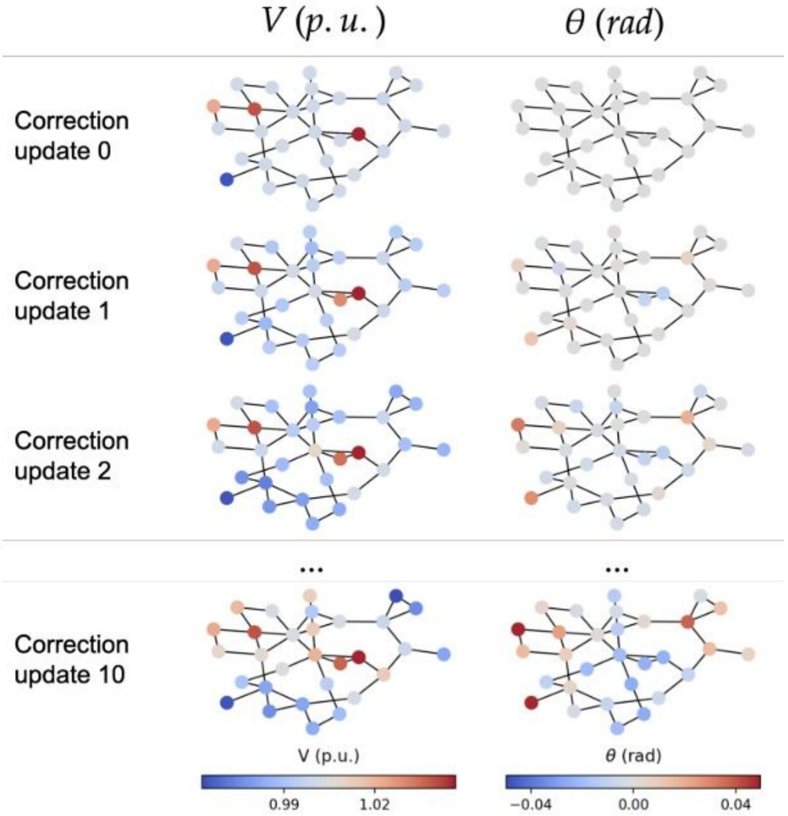
TABLE I
1-DAY AHEAD FORECASTING ERROR OF MODELS TESTED ON UNIVARIATE REAL AND SYNTHETIC DATASETS

Model	PV 4		PV 31		PV 10	
	MSE	MAE	MSE	MAE	MSE	MAE
LSTM-FC	0.0085 ± 0.0002	0.0491 ± 0.0011	0.0243 ± 0.0002	0.0861 ± 0.0008	0.0116 ± 0.0007	0.0598 ± 0.0046
CNN-FC	0.0144 ± 0,0003	0.0673 ± 0,0009	0.0237 ± 0,0003	0.0855 ± 0,0010	0.0109 ± 0,0056	0.0578 ± 0,0019
GNN	**0.0063 ± 0.0002**	**0.0412 ± 0.0003**	**0.0108 ± 0.0004**	**0.0559 ± 0.0013**	**0.0043 ± 0,0008**	**0.0355 ± 0,0003**

A. Verdone, S. Scardapane, M. Panella, 2022. **Multi-site Forecasting of Energy Time Series with Spatio-Temporal Graph Neural Networks**. Accepted at *2022 IEEE WCCI*.

Power systems





Donon, B., Donnot, B., Guyon, I. and Marot, A., 2019. **Graph neural solver for power systems**. In 2019 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

Fully distributed GCNs

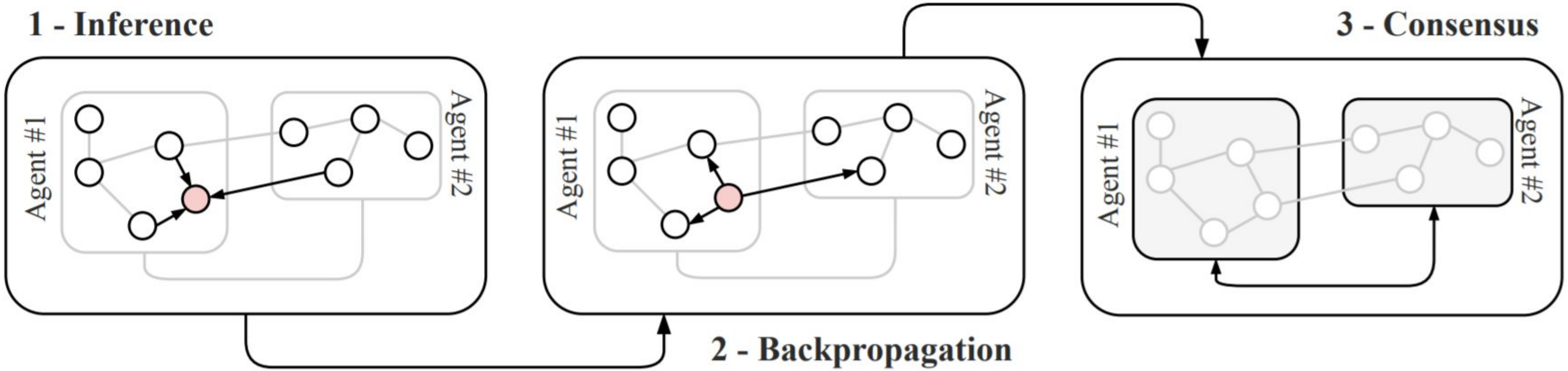
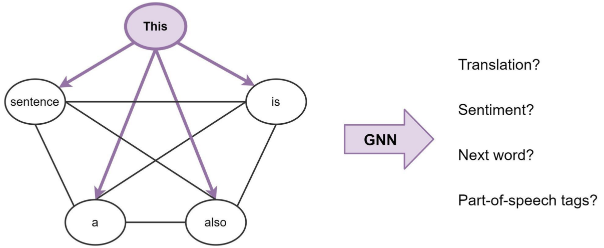


Fig. 1. Illustration of the proposed approach. In step 1, nodes communicate to perform inference. In step 2, a symmetric communication phase is executed to compute local gradients. In step 3, agents exchange local variables to asymptotically reach agreement. For steps 1-2, a representative active node is shown in red. Directed arrows show the flow of messages.

Scardapane, S., Spinelli, I. and Di Lorenzo, P., 2020. **Distributed Training of Graph Convolutional Networks**. *IEEE Transactions on Signal and Information Processing over Networks.*

Graph networks in classic deep learning

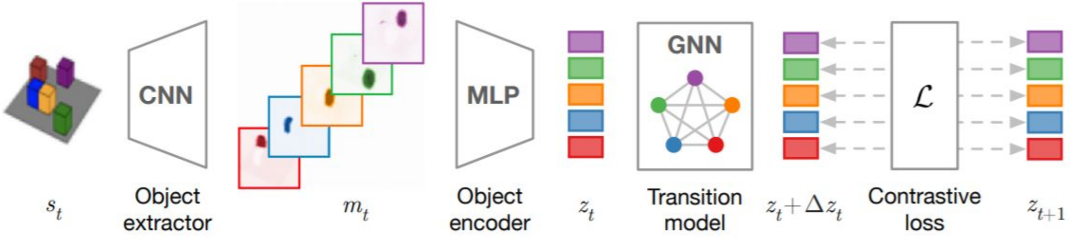


Transformers are basically GNNs on fully-connected graphs!

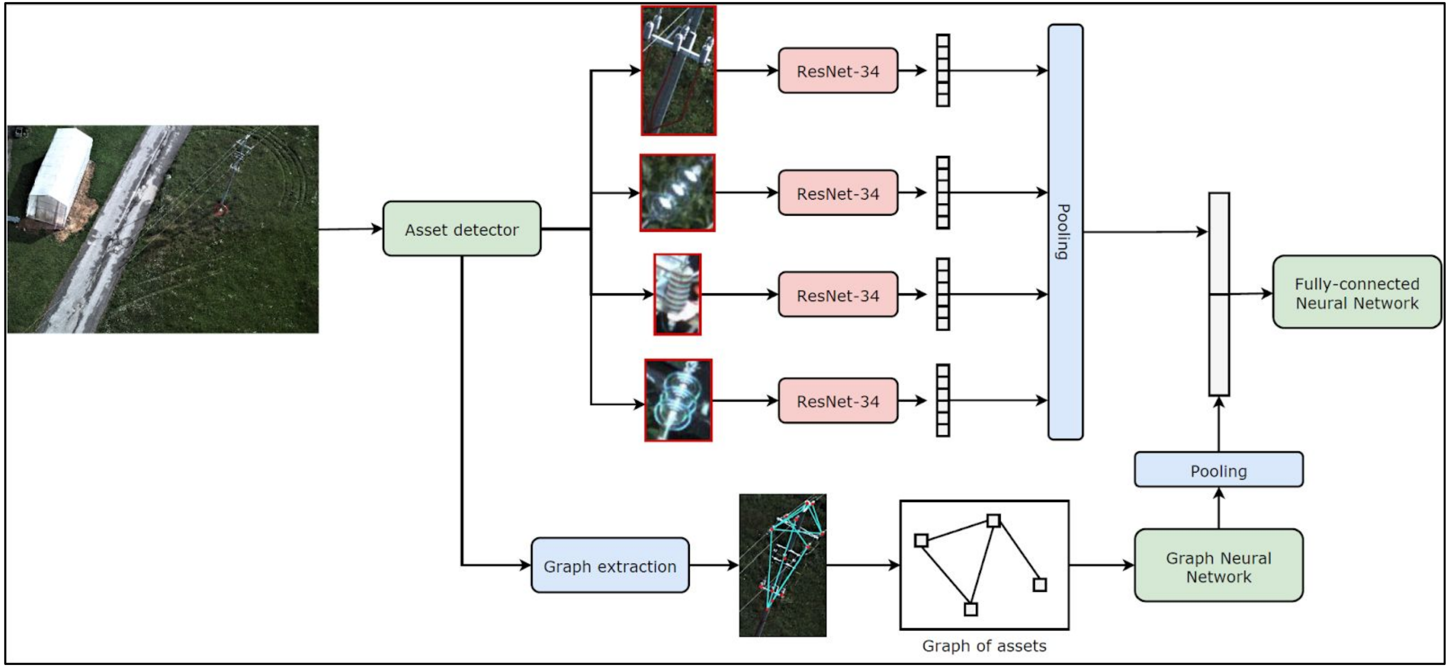
https://thegradient.pub/transformers-are-graph-neural-networks/

[1911.12247] Contrastive Learning of Structured World Models

GNNs can be used to include **relational reasoning** in classical models!



Integrating GNNs in other networks

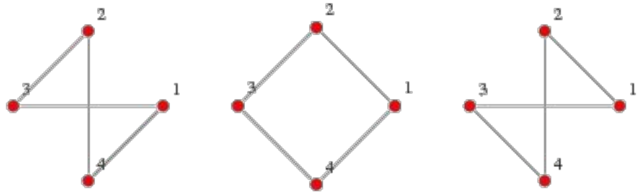


Devoto, A., et al., 2022. **Re-identification of objects from aerial photos with hybrid siamese neural networks**. *IEEE Transactions on Industrial Informatics*, in press.

Graph and learning

Graphs and matrices

Graphs are (represented by) matrices



$$\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Adjacency matrix

$$\mathbf{A} \in \mathbb{R}^{n \times n}$$

n vertices in the graph

Node features

$$\mathbf{X} \in \mathbb{R}^{n \times d}$$

each node has *d* features

Edge features

$$\mathbf{E} \in \mathbb{R}^{e \times f}$$

each edge has *f* features

Adjacency Matrix -- from Wolfram MathWorld

Other interesting matrices

The degree matrix is a diagonal matrix collecting the degrees:

$$D_{ii} = \sum_j A_{ij}$$

We can use the degree matrix to normalize the adjacency matrix:

$$\mathbf{A}_{\text{norm}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$$

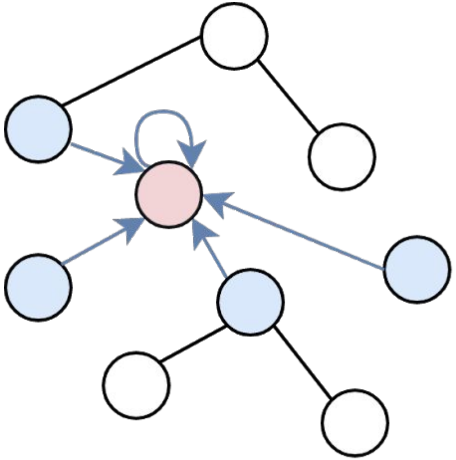
We can also add self-loops to the adjacency matrix: $\mathbf{A} = \mathbf{A} + \mathbf{I}$

In general, we can replace the adjacency with any matrix with a **corresponding sparsity** structure.

Diffusion over a graph

We can view the adjacency matrix as an operator that *diffuses* information across the graph:

$$\lfloor \mathbf{AX} \rfloor_i = \underbrace{\sum_j A_{ij} \mathbf{X}_j}_{\text{sparse sum!}}$$



For a classical adjacency matrix (only 0/1), the above simplifies to a sum, but in general it will be a **weighted sum**.

Laplacian of a graph

The Laplacian is another fundamental graph matrix:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

(Note: we can also build normalized variants by replacing A with any variant seen above.)

Any Laplacian acts as a diffusion operator as above, but its eigen-decomposition is fundamental in a number of disciplines:

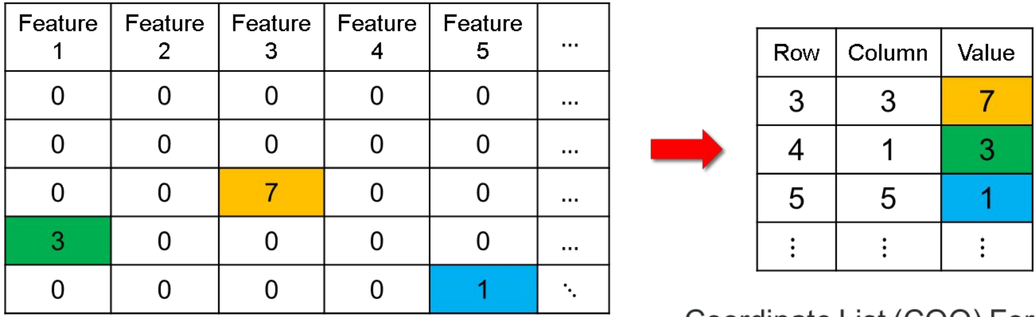
1. **Spectral graph theory**, and associated ML algorithms (e.g., spectral clustering).
2. **Graph signal processing**, where it allows to define equivalents of a Fourier transformation.

Storing the graph matrices

Tip: Working with Sparse Data in SAS - SAS Support Communities

Matrix notation is convenient for describing the operations, but **sparse** matrices have dedicated storage formats in most software.

Coordinate list (COO) and its variants (CSR, CSC) is typical:



Sparse Format

Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	...
0	0	0	0	0	...
0	0	0	0	0	...
0	0	7	0	0	...
3	0	0	0	0	...
0	0	0	0	1	⋱

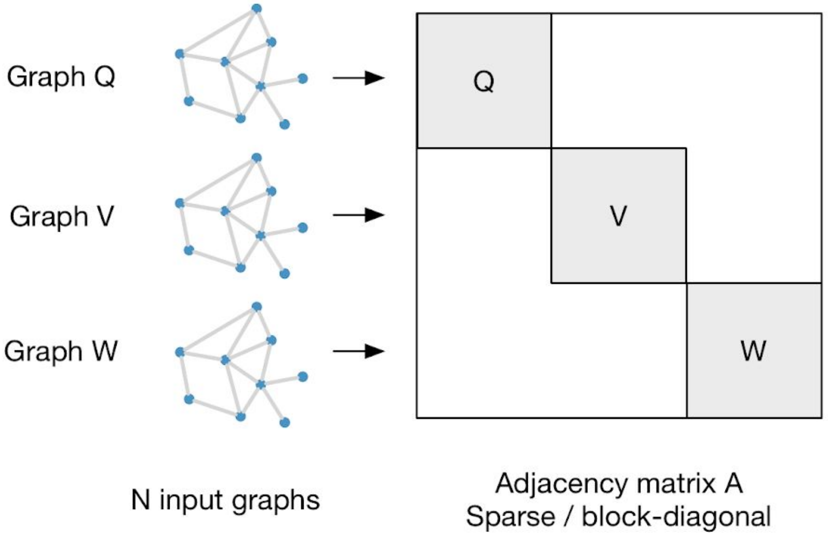
Row	Column	Value
3	3	7
4	1	3
5	5	1
⋮	⋮	⋮

Coordinate List (COO) Format

This also allows highly optimized variants of matrix multiplication.

Handling multiple graphs

We can handle multiple graphs (e.g., mini-batches) by considering a single graph with several disconnected components:

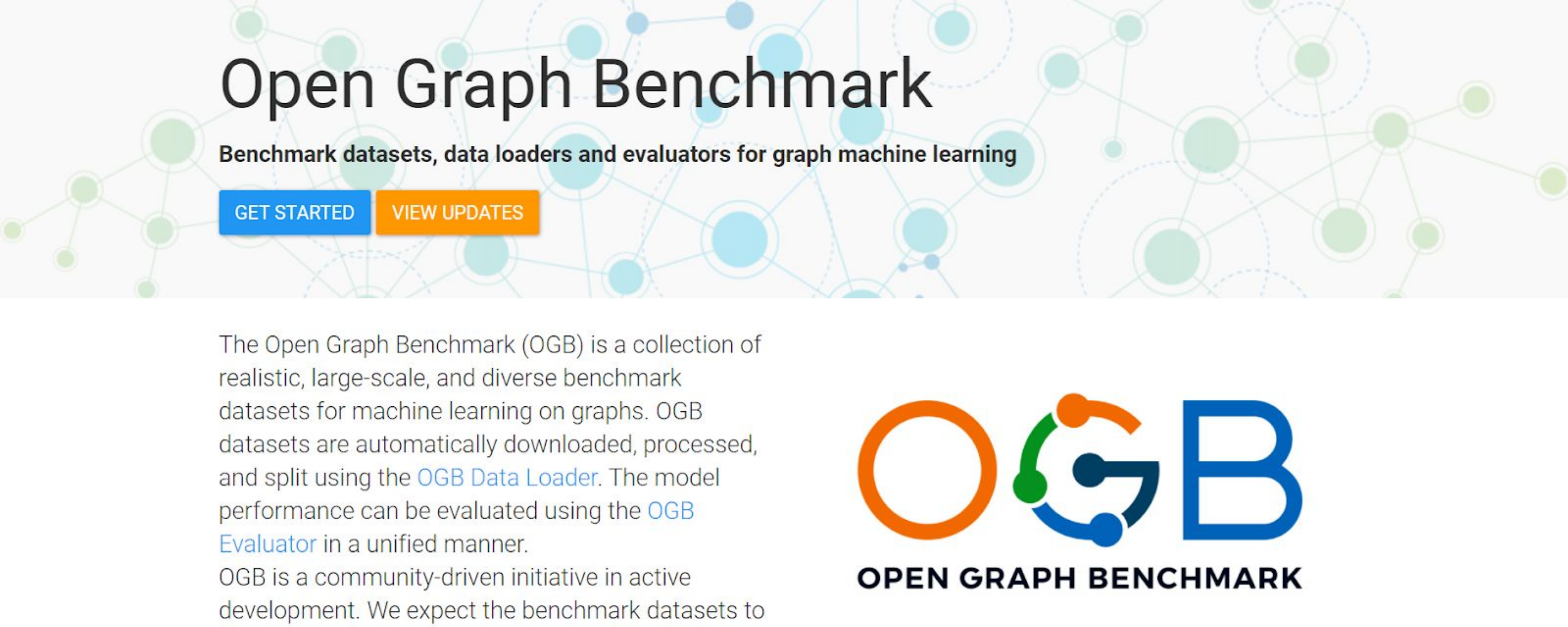


tkipf/gcn: Implementation of Graph Convolutional Networks in TensorFlow

Implementing graph NNs

Software and code

Scaling up to huge graphs



Open Graph Benchmark

Benchmark datasets, data loaders and evaluators for graph machine learning

The Open Graph Benchmark (OGB) is a collection of realistic, large-scale, and diverse benchmark datasets for machine learning on graphs. OGB datasets are automatically downloaded, processed, and split using the OGB Data Loader. The model performance can be evaluated using the OGB Evaluator in a unified manner.
OGB is a community-driven initiative in active development. We expect the benchmark datasets to

Software?

DeepGraphLibrary

Amazon based, production-ready

PyTorch geometric

PyTorch, more research oriented

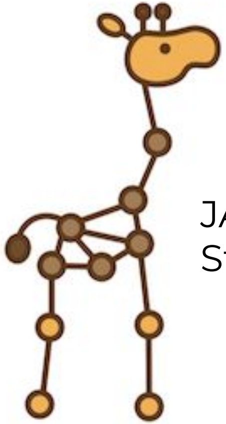
Spektral

Keras-like, TensorFlow 2.0

tensorflow/gnn

TensorFlow GNN is a library to build Graph Neural Networks on the TensorFlow platform.

Alpha release, poorly documented



JAX-based
Strong DeepMind adoption

Notebook time!

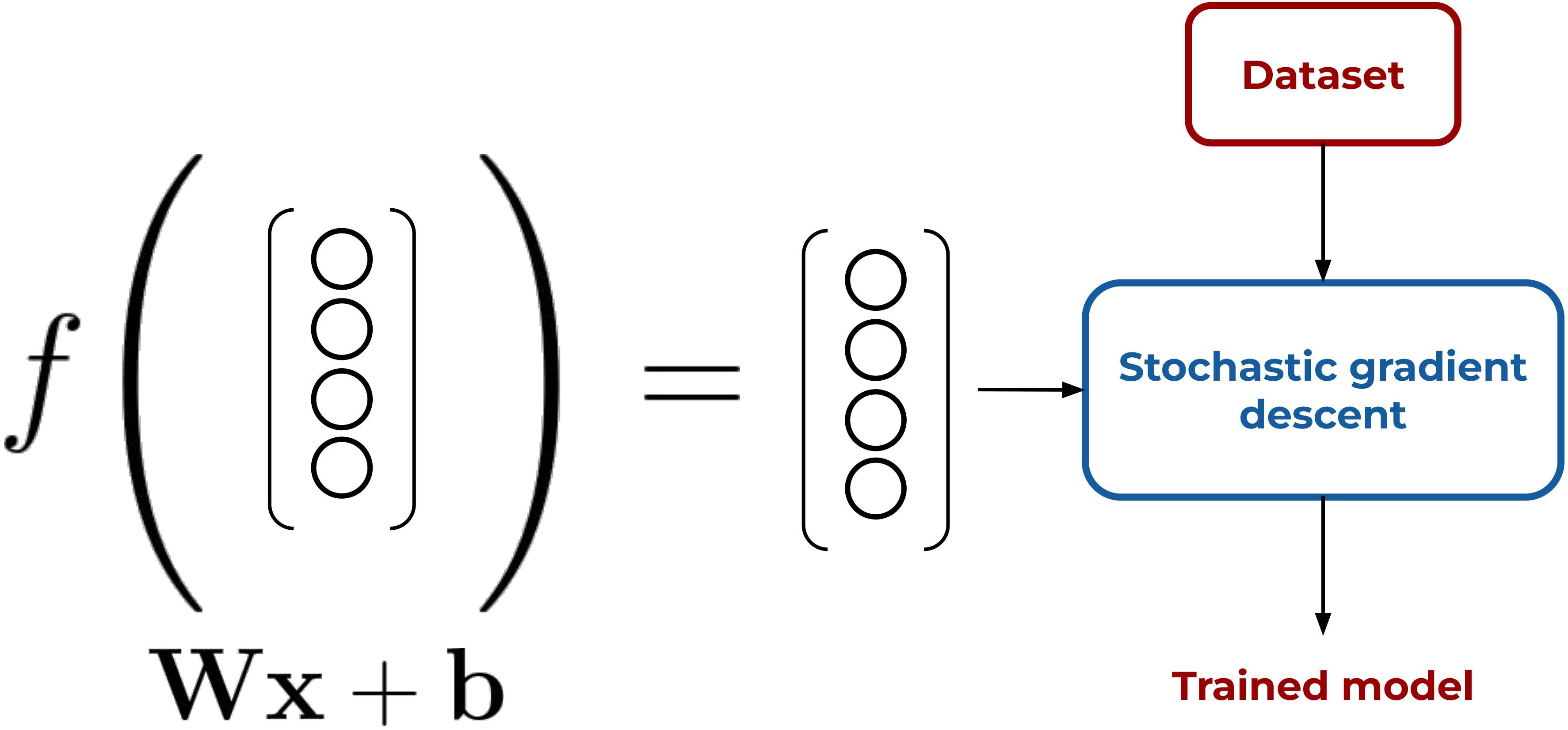
Colab Notebooks and Video Tutorials — pytorch_geometric documentation

https://colab.research.google.com/drive/1nV44NrNqcXC2thU6-zzxnJPnIalo870m?usp=sharing

Building graph layers

Graph convolutional layers

Neural networks in a single slide

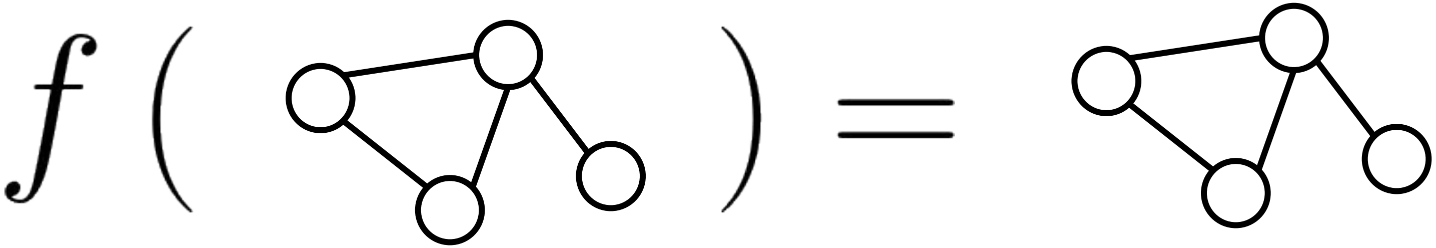


Dataset

Stochastic gradient descent

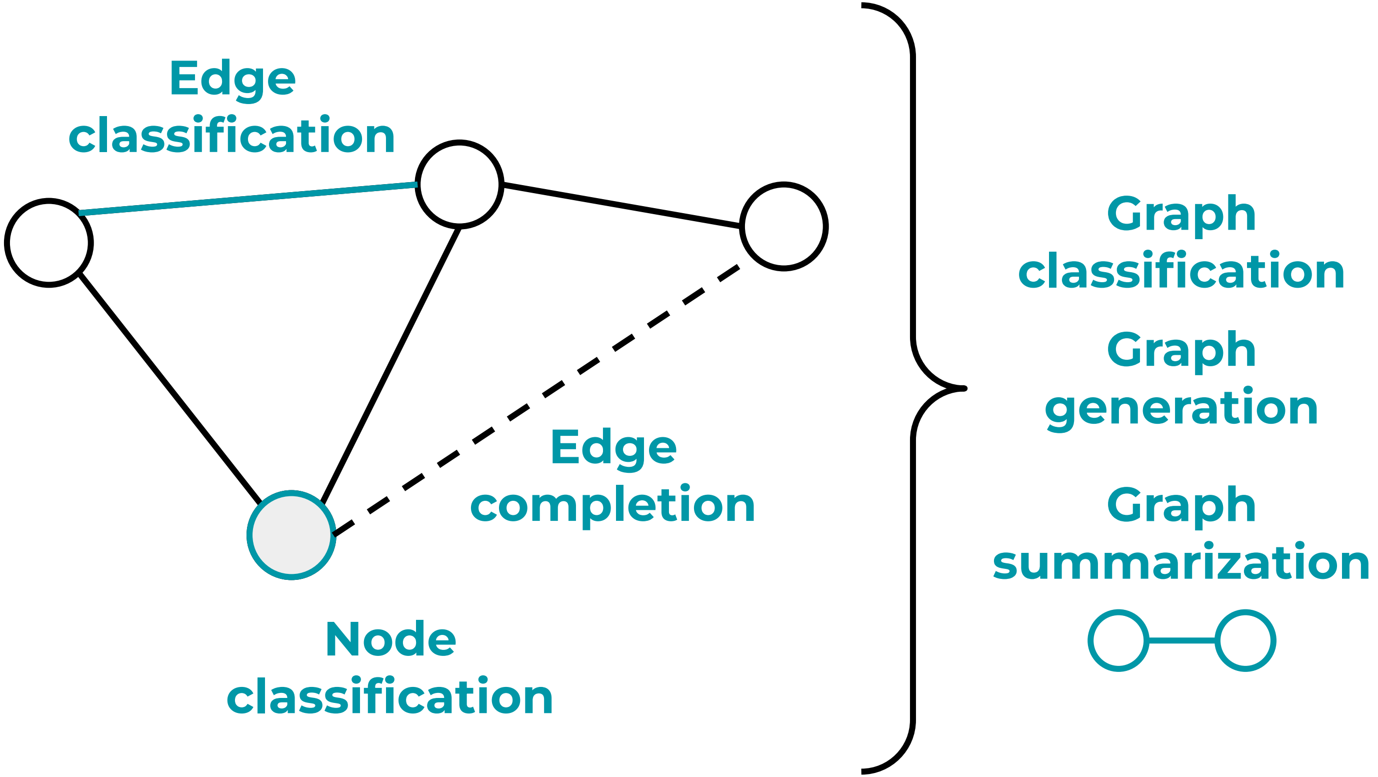
Trained model

How can we learn on a graph?



We want to do deep learning, hence *f* should be **differentiable**, **composable**, **scalable**.

Summary: what can we learn on a graph?



Edge classification

Edge completion

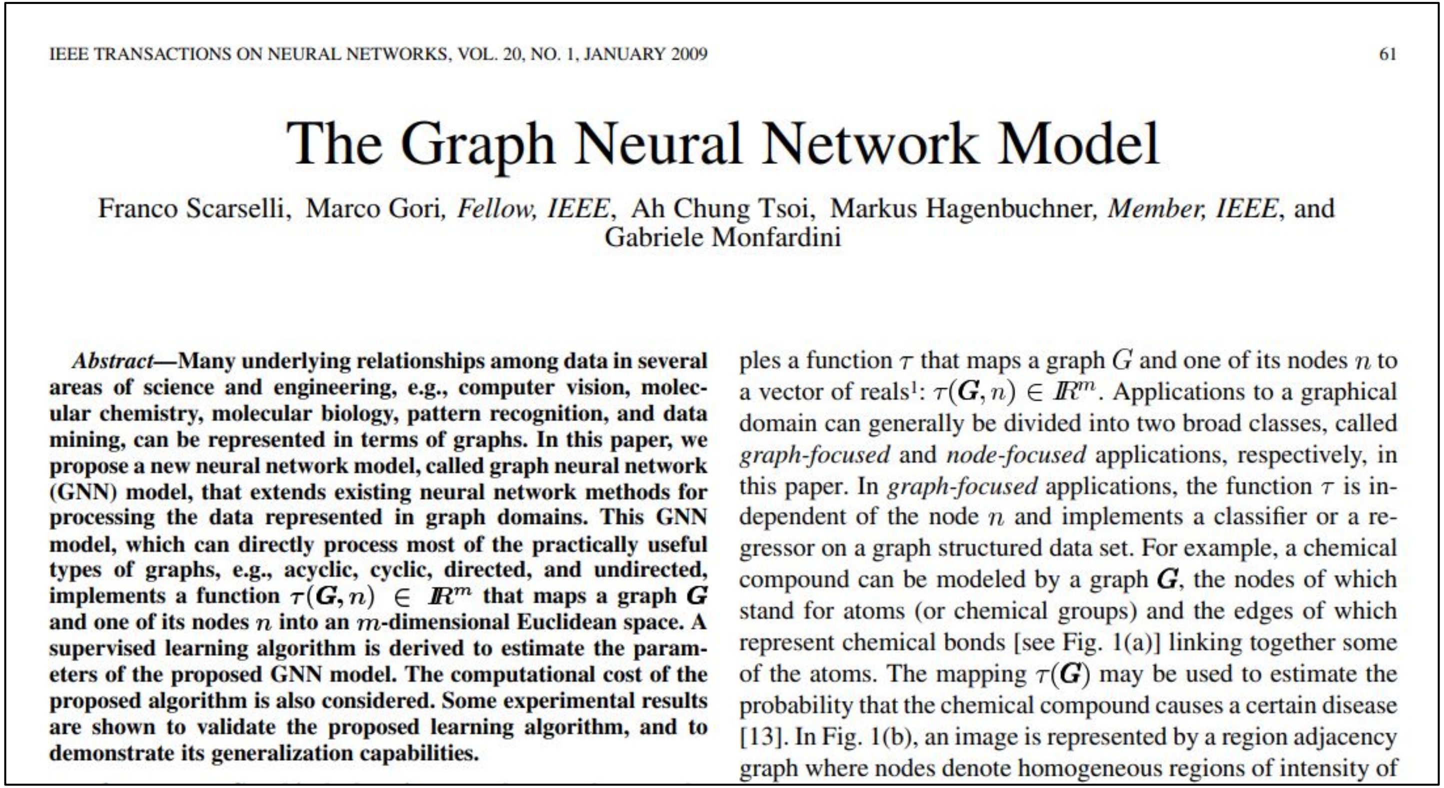
Node classification

Graph classification

Graph generation

Graph summarization

GNNs before deep learning!



IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 20, NO. 1, JANUARY 2009 61

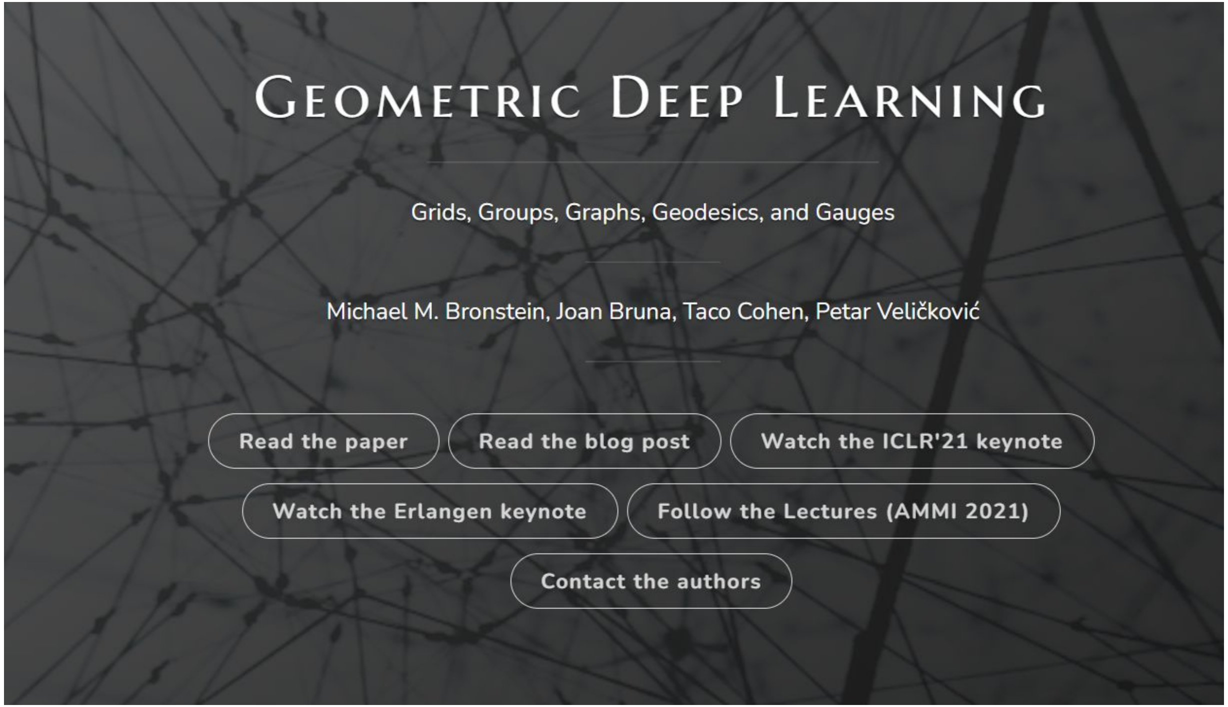
The Graph Neural Network Model

Franco Scarselli, Marco Gori, *Fellow, IEEE*, Ah Chung Tsoi, Markus Hagenbuchner, *Member, IEEE*, and Gabriele Monfardini

***Abstract*—Many underlying relationships among data in several areas of science and engineering, e.g., computer vision, molecular chemistry, molecular biology, pattern recognition, and data mining, can be represented in terms of graphs. In this paper, we propose a new neural network model, called graph neural network (GNN) model, that extends existing neural network methods for processing the data represented in graph domains. This GNN model, which can directly process most of the practically useful types of graphs, e.g., acyclic, cyclic, directed, and undirected, implements a function $\tau(\boldsymbol{G}, n) \in \mathbb{R}^m$ that maps a graph $\boldsymbol{G}$ and one of its nodes n into an m-dimensional Euclidean space. A supervised learning algorithm is derived to estimate the parameters of the proposed GNN model. The computational cost of the proposed algorithm is also considered. Some experimental results are shown to validate the proposed learning algorithm, and to demonstrate its generalization capabilities.**

ples a function τ that maps a graph G and one of its nodes n to a vector of reals[1]: $\tau(\boldsymbol{G}, n) \in \mathbb{R}^m$. Applications to a graphical domain can generally be divided into two broad classes, called *graph-focused* and *node-focused* applications, respectively, in this paper. In *graph-focused* applications, the function τ is independent of the node n and implements a classifier or a regressor on a graph structured data set. For example, a chemical compound can be modeled by a graph $\boldsymbol{G}$, the nodes of which stand for atoms (or chemical groups) and the edges of which represent chemical bonds [see Fig. 1(a)] linking together some of the atoms. The mapping $\tau(\boldsymbol{G})$ may be used to estimate the probability that the chemical compound causes a certain disease [13]. In Fig. 1(b), an image is represented by a region adjacency graph where nodes denote homogeneous regions of intensity of

Geometric deep learning



Geometric Deep Learning

Grids, Groups, Graphs, Geodesics, and Gauges

Michael M. Bronstein, Joan Bruna, Taco Cohen, Petar Veličković

Read the paper
Read the blog post
Watch the ICLR'21 keynote
Watch the Erlangen keynote
Follow the Lectures (AMMI 2021)
Contact the authors

Bronstein, M.M., Bruna, J., Cohen, T. and Veličković, P., 2021. **Geometric deep learning: Grids, groups, graphs, geodesics, and gauges**. *arXiv preprint arXiv:2104.13478*.

A zoo of techniques...

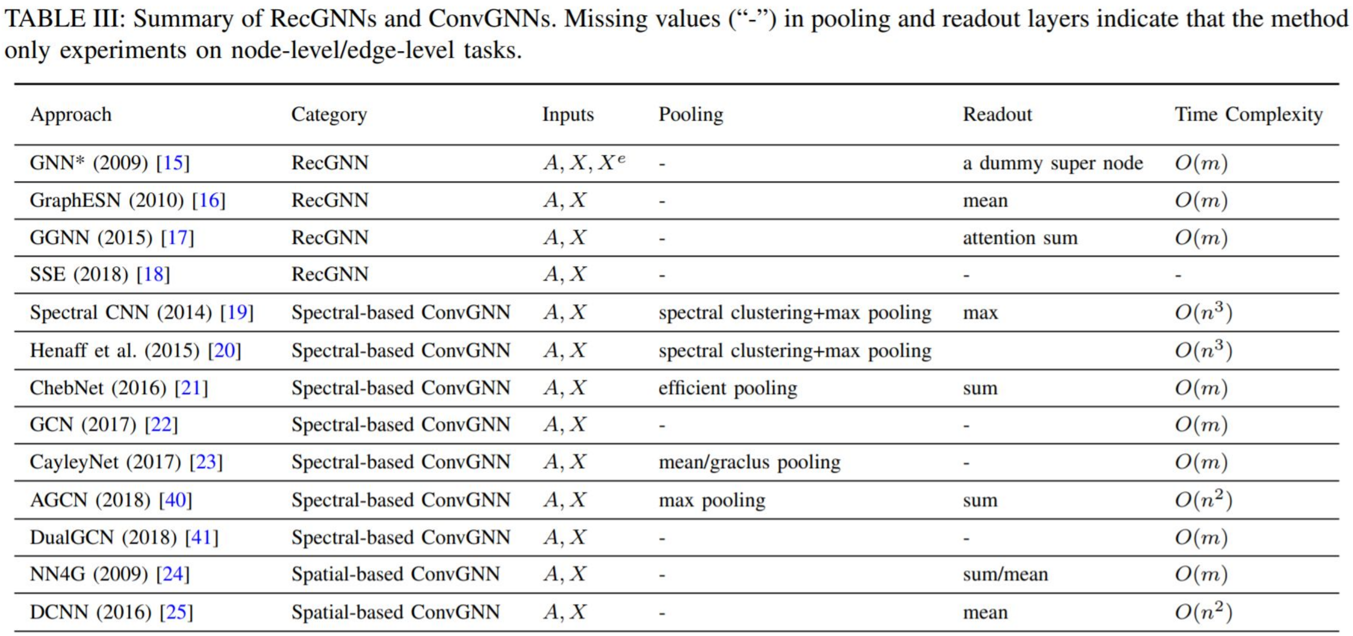
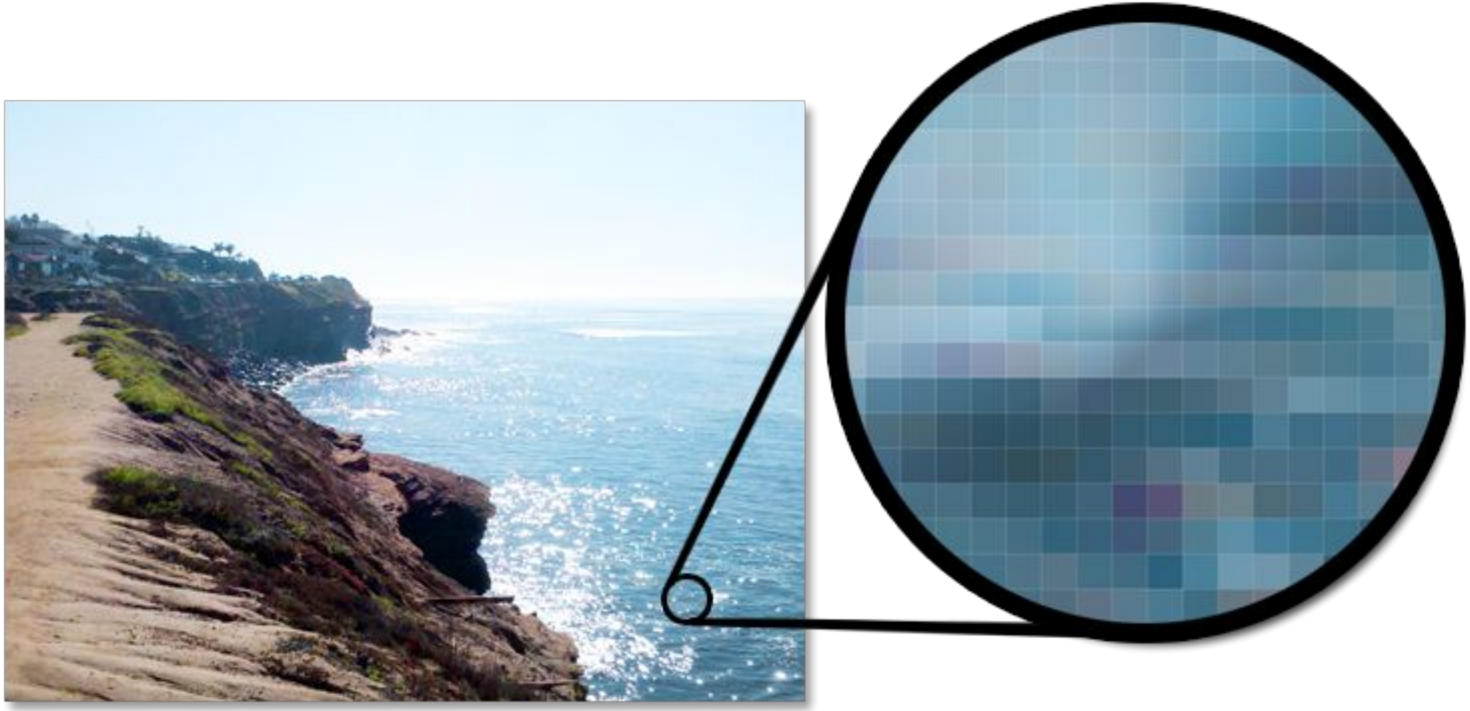


TABLE III: Summary of RecGNNs and ConvGNNs. Missing values ("-") in pooling and readout layers indicate that the method only experiments on node-level/edge-level tasks.

Approach	Category	Inputs	Pooling	Readout	Time Complexity
GNN* (2009) [15]	RecGNN	A, X, X^e	-	a dummy super node	$O(m)$
GraphESN (2010) [16]	RecGNN	A, X	-	mean	$O(m)$
GGNN (2015) [17]	RecGNN	A, X	-	attention sum	$O(m)$
SSE (2018) [18]	RecGNN	A, X	-	-	-
Spectral CNN (2014) [19]	Spectral-based ConvGNN	A, X	spectral clustering+max pooling	max	$O(n^3)$
Henaff et al. (2015) [20]	Spectral-based ConvGNN	A, X	spectral clustering+max pooling		$O(n^3)$
ChebNet (2016) [21]	Spectral-based ConvGNN	A, X	efficient pooling	sum	$O(m)$
GCN (2017) [22]	Spectral-based ConvGNN	A, X	-	-	$O(m)$
CayleyNet (2017) [23]	Spectral-based ConvGNN	A, X	mean/graclus pooling	-	$O(m)$
AGCN (2018) [40]	Spectral-based ConvGNN	A, X	max pooling	sum	$O(n^2)$
DualGCN (2018) [41]	Spectral-based ConvGNN	A, X	-	-	$O(m)$
NN4G (2009) [24]	Spatial-based ConvGNN	A, X	-	sum/mean	$O(m)$
DCNN (2016) [25]	Spatial-based ConvGNN	A, X	-	mean	$O(n^2)$

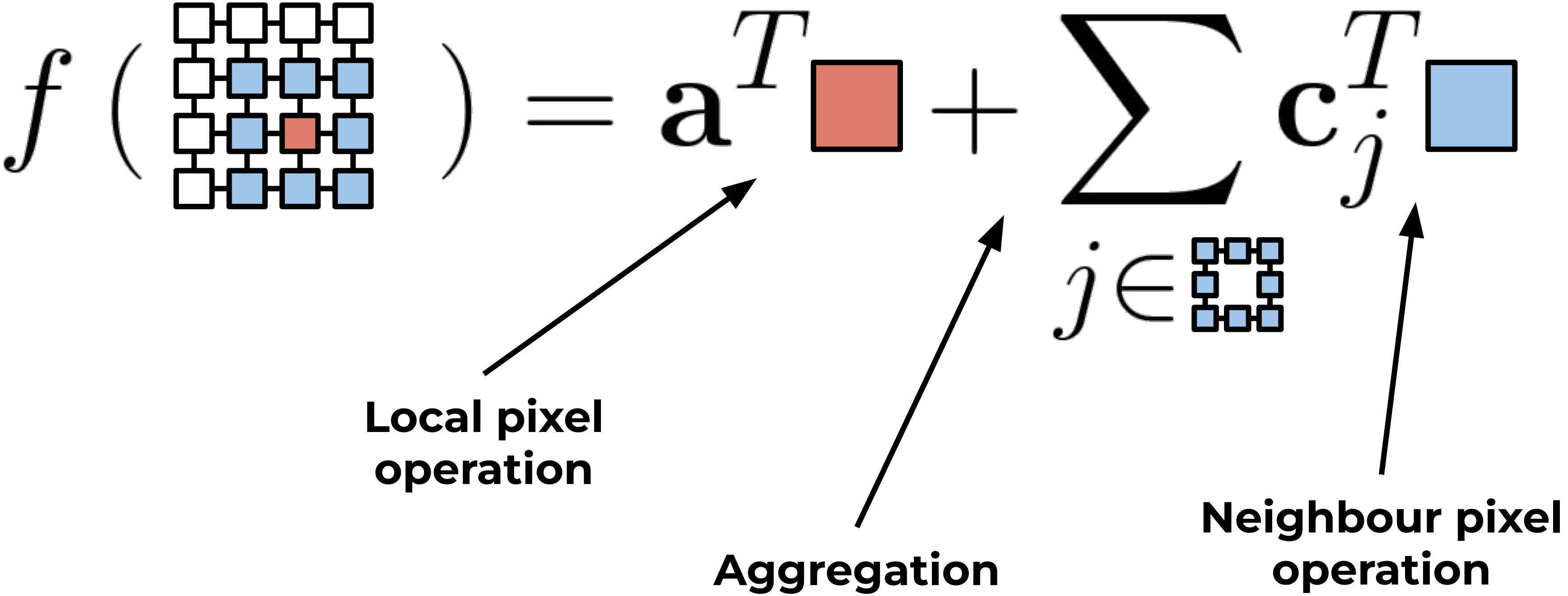
Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. and Yu, P.S., 2019. **A comprehensive survey on graph neural networks**. arXiv preprint arXiv:1901.00596.

Deep learning is about leveraging structure



Generate cross-stitch patterns from any image.

Image convolutions



Local pixel operation

Aggregation

Neighbour pixel operation

Images vs. graphs

How much of the structure of an image do we find in graphs?

✔️ **Locality** (neighbourhood).

❌ Fixed number of neighbours.

❌ Neighbours have a definite ordering.

Permutation equivariance

Consider a generic permutation matrix of dimension *n*:

$$\underbrace{\begin{bmatrix}0 & 0 & 1\\ 1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}}_{\mathbf{P}}\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\\ \mathbf{x}_3\end{bmatrix}=\begin{bmatrix}\mathbf{x}_2\\ \mathbf{x}_1\\ \mathbf{x}_3\end{bmatrix}$$

Permuting the nodes of a graph results in an equivalent graph (**isomorphism**):

$$(\mathbf{PX}, \mathbf{PAP}^\top) \sim (\mathbf{X}, \mathbf{A})$$

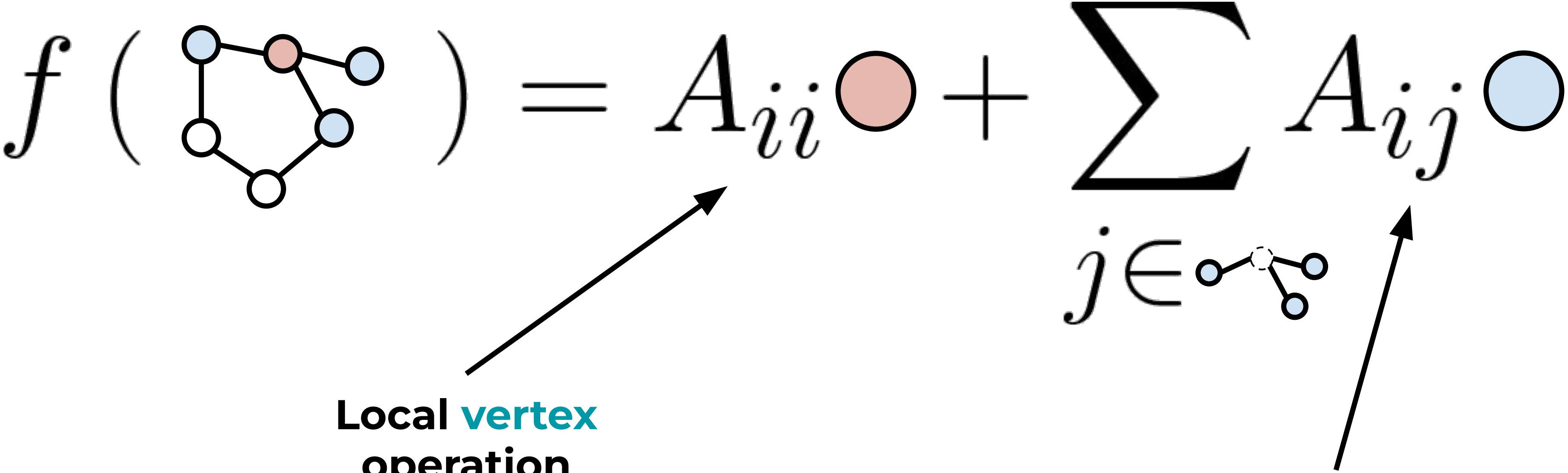
Permutation equivariance (2)

Any graph layer must possess a property called **permutation equivariance**:

$$\underbrace{f(\mathbf{PX}, \mathbf{PAP}^\top)}_{\text{permuted input}} = \overbrace{\mathbf{P}f(\mathbf{X}, \mathbf{A})}^{\text{permuted output}}$$

Running the layer with a different node ordering should modify the output ordering **only**.

Putting everything together (graph convolutions)



Local **vertex** **operation**

We use the **adjacency matrix** **(or similar)**

Note: the mixing coefficients are no more learnable!
Here, only local operations can be trainable.

Graph convolutions

Writing it out explicitly for a single node:

$$\lfloor \mathbf{H} \rfloor_i = \sum_j A_{ij} f(\mathbf{X}_j) + A_{ii} g(\mathbf{X}_i)$$

In the simplest case (this is a classical **graph convolutional layer**):

$$f(\mathbf{x}) = g(\mathbf{x}) = \mathbf{W}\mathbf{x}$$

Kipf, T.N. and Welling, M., 2016. **Semi-supervised classification with graph convolutional networks**. arXiv preprint arXiv:1609.02907.

Properties of a graph convolutional layer

The layer can be understood as:

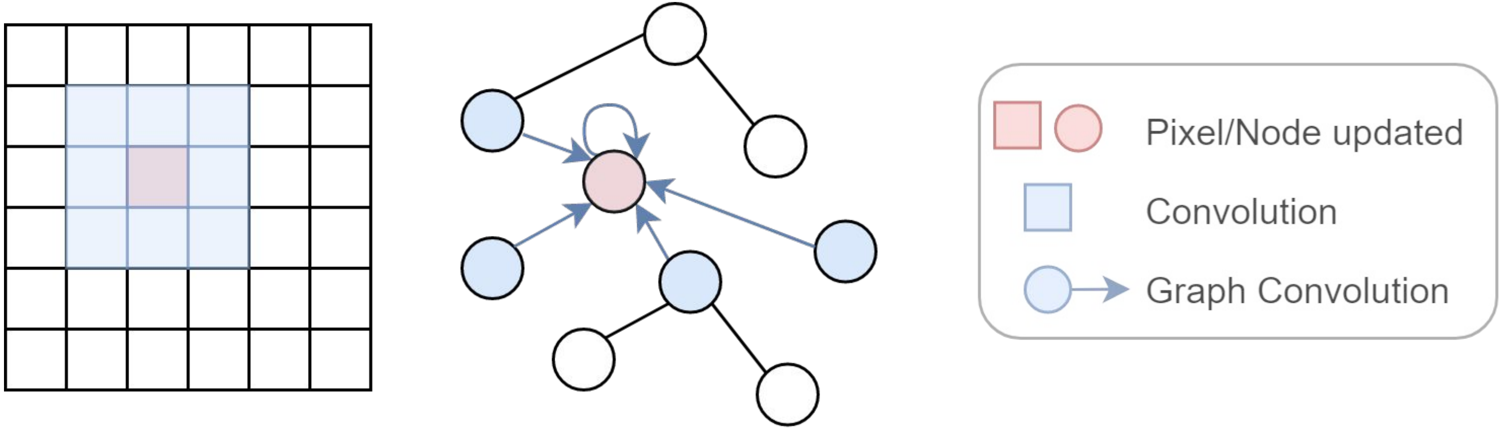
1. Applying a local operation at every node;
2. **Aggregating** the updated embeddings depending on the graph topology.

(If you are curious: it can be understood as a linear filter in the spectral domain given by the Graph Fourier transform.)

We can write it out compactly as:

$$\mathbf{H} = \mathbf{A}\mathbf{X}\mathbf{W}$$

Graph convolution visualized

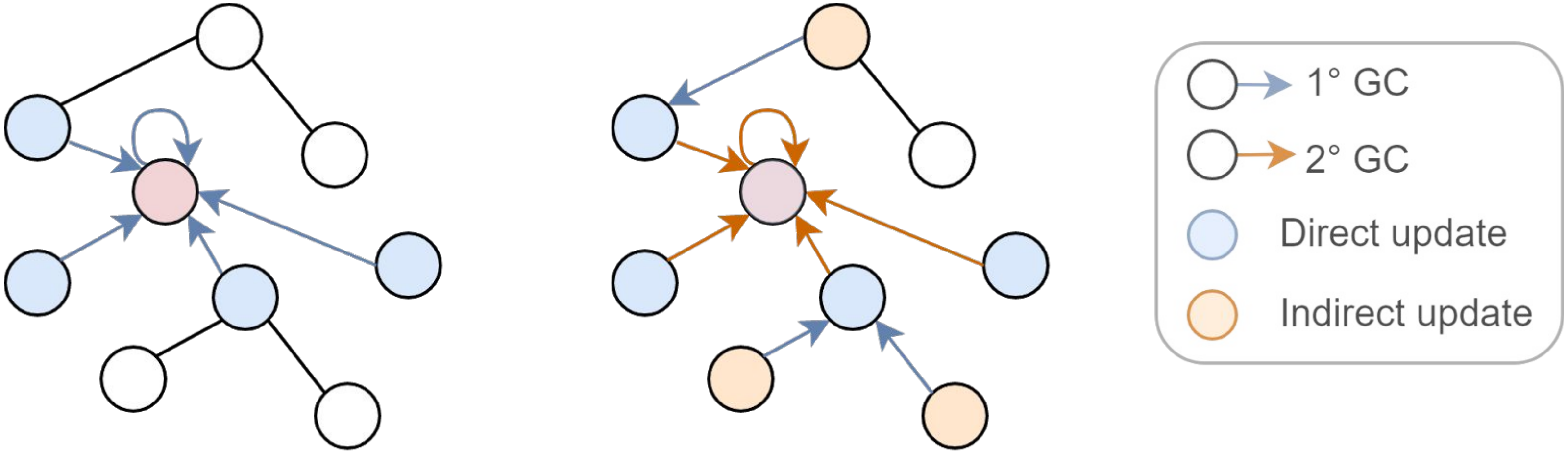


[Seminar on Graph Neural Networks | Indro Spinelli]

Building graph *networks*

Stacking graph layers

Stacking graph convolutional layers



Performing multiple updates increases the "**receptive field**" of each node.

https://spindro.github.io/post/gnn/

Building deep graph networks

Stacking graph layers works similarly to standard deep networks, since the topology of the graph is unchanged:

$$\mathbf{H} = \phi\left(\mathbf{A}\underbrace{\phi\left(\mathbf{AXW}_1\right)}_{\text{layer 1}}\mathbf{W}_2\right)$$

Deep graph networks may suffer from **oversmoothing** (all representations collapse to the same). Some specialized solutions exist (e.g., PairNorm regularization).

Pooling

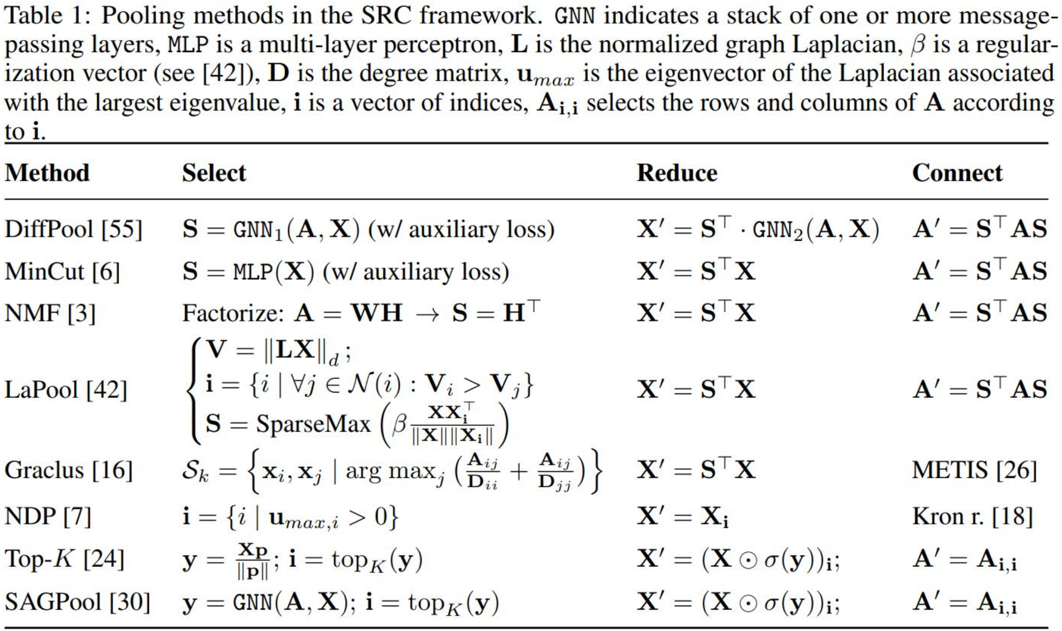
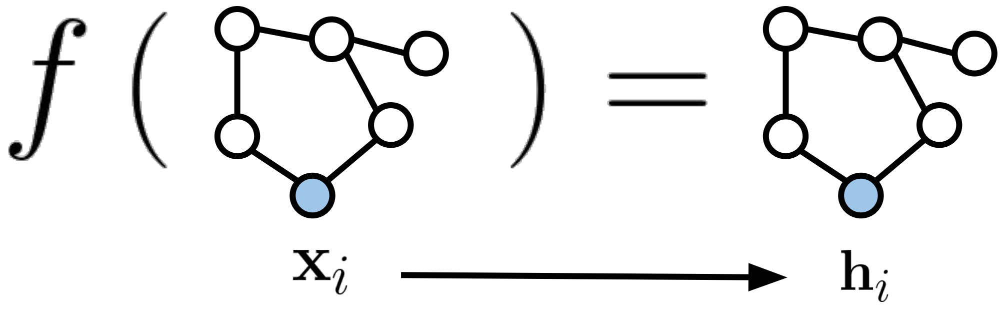


Table 1: Pooling methods in the SRC framework. GNN indicates a stack of one or more message-passing layers, MLP is a multi-layer perceptron, $\mathbf{L}$ is the normalized graph Laplacian, β is a regularization vector (see [42]), $\mathbf{D}$ is the degree matrix, $\mathbf{u}_{max}$ is the eigenvector of the Laplacian associated with the largest eigenvalue, $\mathbf{i}$ is a vector of indices, $\mathbf{A}_{\mathbf{i},\mathbf{i}}$ selects the rows and columns of $\mathbf{A}$ according to $\mathbf{i}$.

Method	**Select**	**Reduce**	**Connect**
DiffPool [55]	$\mathbf{S} = \texttt{GNN}_1(\mathbf{A}, \mathbf{X})$ (w/ auxiliary loss)	$\mathbf{X}' = \mathbf{S}^\top \cdot \texttt{GNN}_2(\mathbf{A}, \mathbf{X})$	$\mathbf{A}' = \mathbf{S}^\top \mathbf{A} \mathbf{S}$
MinCut [6]	$\mathbf{S} = \texttt{MLP}(\mathbf{X})$ (w/ auxiliary loss)	$\mathbf{X}' = \mathbf{S}^\top \mathbf{X}$	$\mathbf{A}' = \mathbf{S}^\top \mathbf{A} \mathbf{S}$
NMF [3]	Factorize: $\mathbf{A} = \mathbf{W}\mathbf{H} \to \mathbf{S} = \mathbf{H}^\top$	$\mathbf{X}' = \mathbf{S}^\top \mathbf{X}$	$\mathbf{A}' = \mathbf{S}^\top \mathbf{A} \mathbf{S}$
LaPool [42]	$\begin{cases} \mathbf{V} = \lVert \mathbf{L}\mathbf{X} \rVert_d ; \\ \mathbf{i} = \{ i \mid \forall j \in \mathcal{N}(i) : \mathbf{V}_i > \mathbf{V}_j \} \\ \mathbf{S} = \text{SparseMax}\left(\beta \frac{\mathbf{X}\mathbf{X}_\mathbf{i}^\top}{\lVert \mathbf{X} \rVert \lVert \mathbf{X}_\mathbf{i} \rVert} \right) \end{cases}$	$\mathbf{X}' = \mathbf{S}^\top \mathbf{X}$	$\mathbf{A}' = \mathbf{S}^\top \mathbf{A} \mathbf{S}$
Graclus [16]	$\mathcal{S}_k = \left\{ \mathbf{x}_i, \mathbf{x}_j \mid \arg\max_j \left(\frac{\mathbf{A}_{ij}}{\mathbf{D}_{ii}} + \frac{\mathbf{A}_{ij}}{\mathbf{D}_{jj}} \right) \right\}$	$\mathbf{X}' = \mathbf{S}^\top \mathbf{X}$	METIS [26]
NDP [7]	$\mathbf{i} = \{ i \mid \mathbf{u}_{max,i} > 0 \}$	$\mathbf{X}' = \mathbf{X}_\mathbf{i}$	Kron r. [18]
Top-K [24]	$\mathbf{y} = \frac{\mathbf{X}\mathbf{p}}{\lVert \mathbf{p} \rVert}$; $\mathbf{i} = \text{top}_K(\mathbf{y})$	$\mathbf{X}' = (\mathbf{X} \odot \sigma(\mathbf{y}))_\mathbf{i}$;	$\mathbf{A}' = \mathbf{A}_{\mathbf{i},\mathbf{i}}$
SAGPool [30]	$\mathbf{y} = \texttt{GNN}(\mathbf{A}, \mathbf{X})$; $\mathbf{i} = \text{top}_K(\mathbf{y})$	$\mathbf{X}' = (\mathbf{X} \odot \sigma(\mathbf{y}))_\mathbf{i}$;	$\mathbf{A}' = \mathbf{A}_{\mathbf{i},\mathbf{i}}$

Grattarola, D., Zambon, D., Bianchi, F.M. and Alippi, C., 2021. **Understanding pooling in graph neural networks**. *arXiv preprint arXiv:2110.05292*.

Tackling multiple tasks



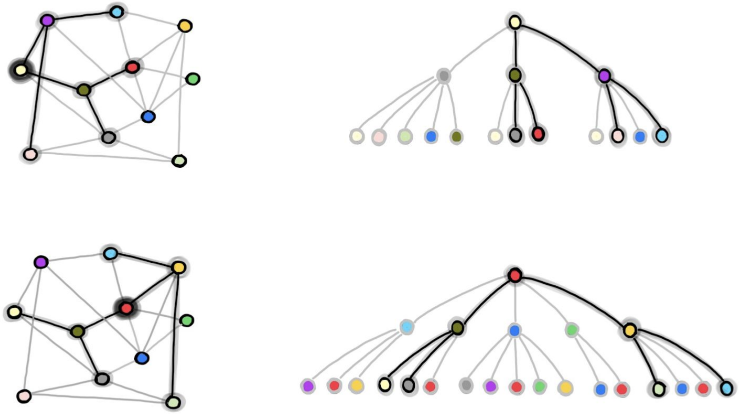
1. Node classification: $\mathrm{sotfmax}(\mathbf{h}_i)$
2. Edge classification: $\mathrm{sotfmax}(\mathbf{h}_i^T\mathbf{h}_j)$
3. Graph classification: $\mathrm{sotfmax}\left(\frac{1}{N}\sum_i \mathbf{h}_i\right)$

Mini-batching in graph networks

There are two types of mini-batching for graph networks:

1. For graph-level tasks, we can create a mini-batch of several graphs.

2. For node/edge-level tasks, we can create a mini-batch by **sampling** nodes and edges from a larger graph.



Simple scalable graph neural networks

Notebook time!

Colab Notebooks and Video Tutorials — pytorch_geometric documentation

https://colab.research.google.com/drive/1nV44NrNqcXC2thU6-zzxnJPnIalo870m?usp=sharing

Beyond graph convolutions

Message-passing graph layers

Revisiting the GCN layer

We can write down the GCN layer in a slightly more general form:

$$\mathbf{H}_i = \mathrm{Aggregate}\left(\{\phi(\mathbf{X}_j) | j \in \mathcal{N}_i\}\right)$$

For each neighbour we build an update, which is the GCN case is weighted by the corresponding element of the adjacency matrix.

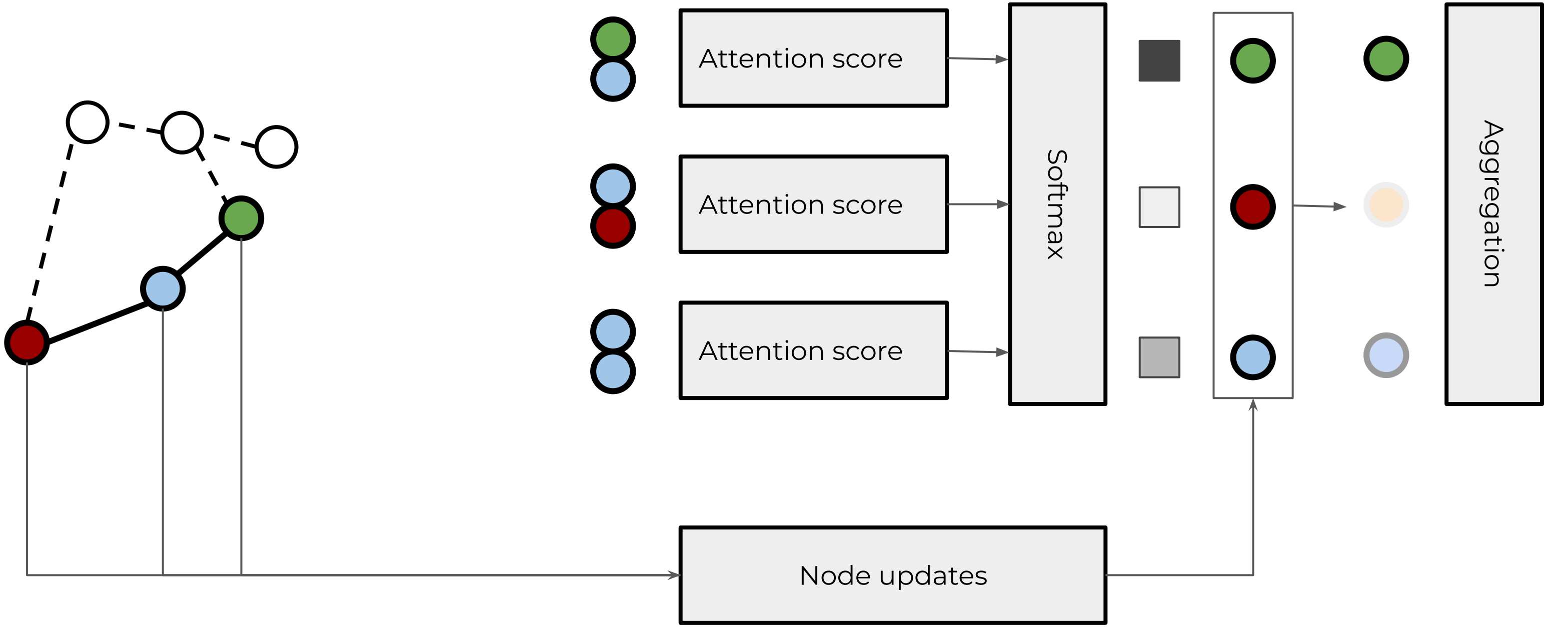
Attention over nodes

The biggest limit of graph convolutional layers is that the weight node *i* gives to node *j* is fixed.

In principle, it is possible that one neighbour is more important than another, and we would like to learn this.

This can be achieved with an **attention mechanism**.

Visualizing graph attention (for node ●)



Brody, S., Alon, U. and Yahav, E., 2021. **How attentive are graph attention networks?**. *arXiv preprint arXiv:2105.14491*.

A more general setup (2)

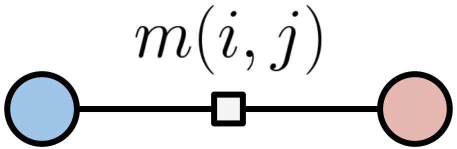
More in detail, in a **graph attention layer**, we compute messages using an attention mechanism:

$$\psi(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{a}^\top \mathrm{LeakyReLU}\left(\mathbf{W}\left[\mathbf{x}_i \,\|\, \mathbf{x}_j\right]\right)$$

$$\mathbf{m}_i = \sum_j \mathrm{softmax}_j(\mathbf{m}_{ij})(\mathbf{W}\mathbf{x}_j)$$

Brody, S., Alon, U. and Yahav, E., 2021. **How attentive are graph attention networks?**. *arXiv preprint arXiv:2105.14491*.

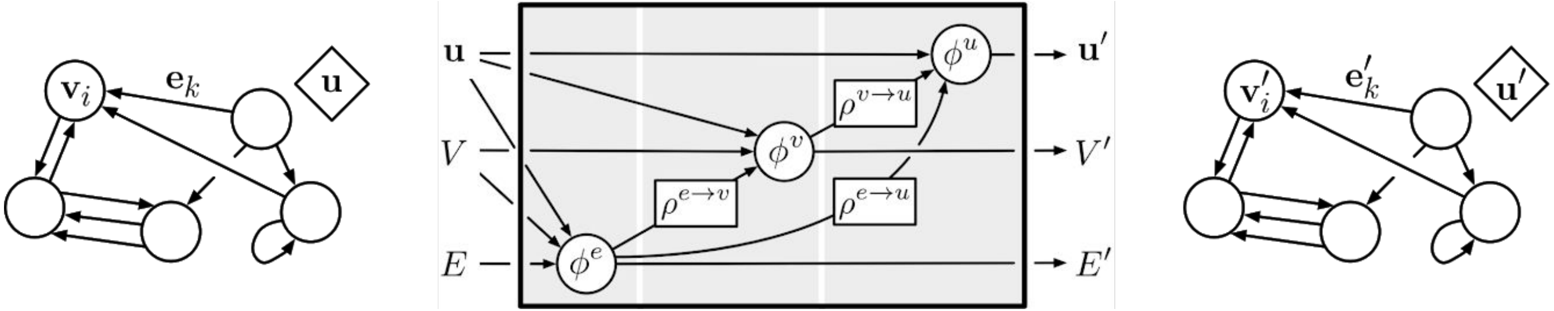
Message-passing neural networks



Instead of using directly the adjacency matrix, nodes can exchange **messages** with several mechanisms (e.g., attention models).

Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O. and Dahl, G.E., 2017, July. **Neural message passing for quantum chemistry**. In *International Conference on Machine Learning* (pp. 1263-1272). PMLR.

A more general setup



Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R. and Gulcehre, C., 2018. **Relational inductive biases, deep learning, and graph networks**. arXiv preprint arXiv:1806.01261.

Notebook time!

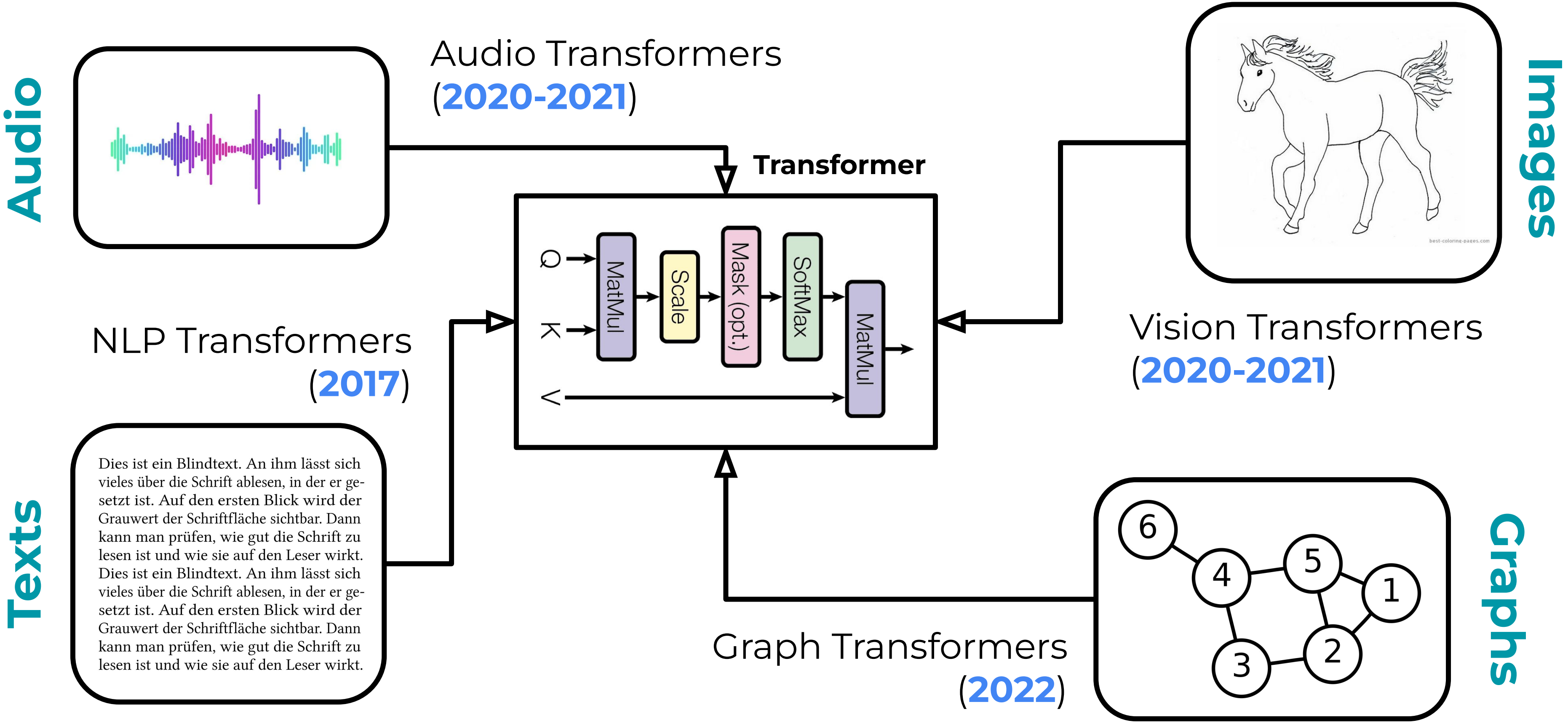
Colab Notebooks and Video Tutorials — pytorch_geometric documentation

https://colab.research.google.com/drive/1nV44NrNqcXC2thU6-zzxnJPnIalo870m?usp=sharing

Beyond message-passing

Transformers as universal neural models

The Transformers revolution



Audio

Audio Transformers (2020-2021)

Transformer

Images

Vision Transformers (2020-2021)

NLP Transformers (2017)

Texts

Dies ist ein Blindtext. An ihm lässt sich vieles über die Schrift ablesen, in der er gesetzt ist. Auf den ersten Blick wird der Grauwert der Schriftfläche sichtbar. Dann kann man prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt. Dies ist ein Blindtext. An ihm lässt sich vieles über die Schrift ablesen, in der er gesetzt ist. Auf den ersten Blick wird der Grauwert der Schriftfläche sichtbar. Dann kann man prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt.

Graph Transformers (2022)

Graphs

Scaling laws for Vision Transformers

[2106.04560] Scaling Vision Transformers

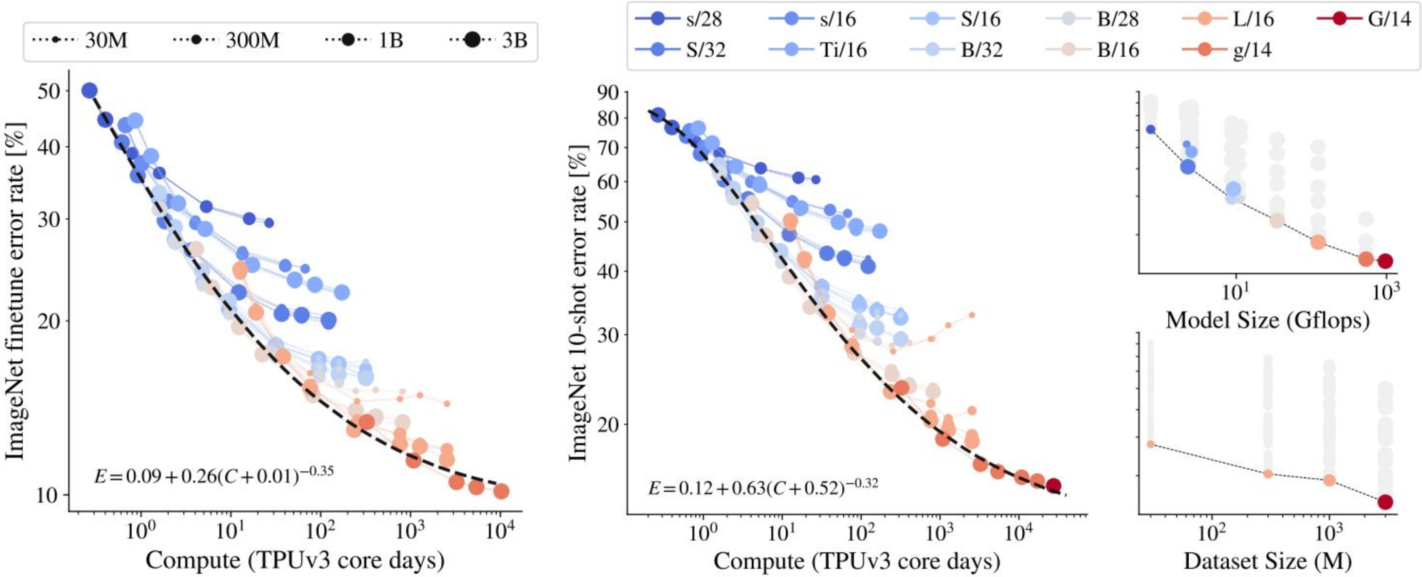
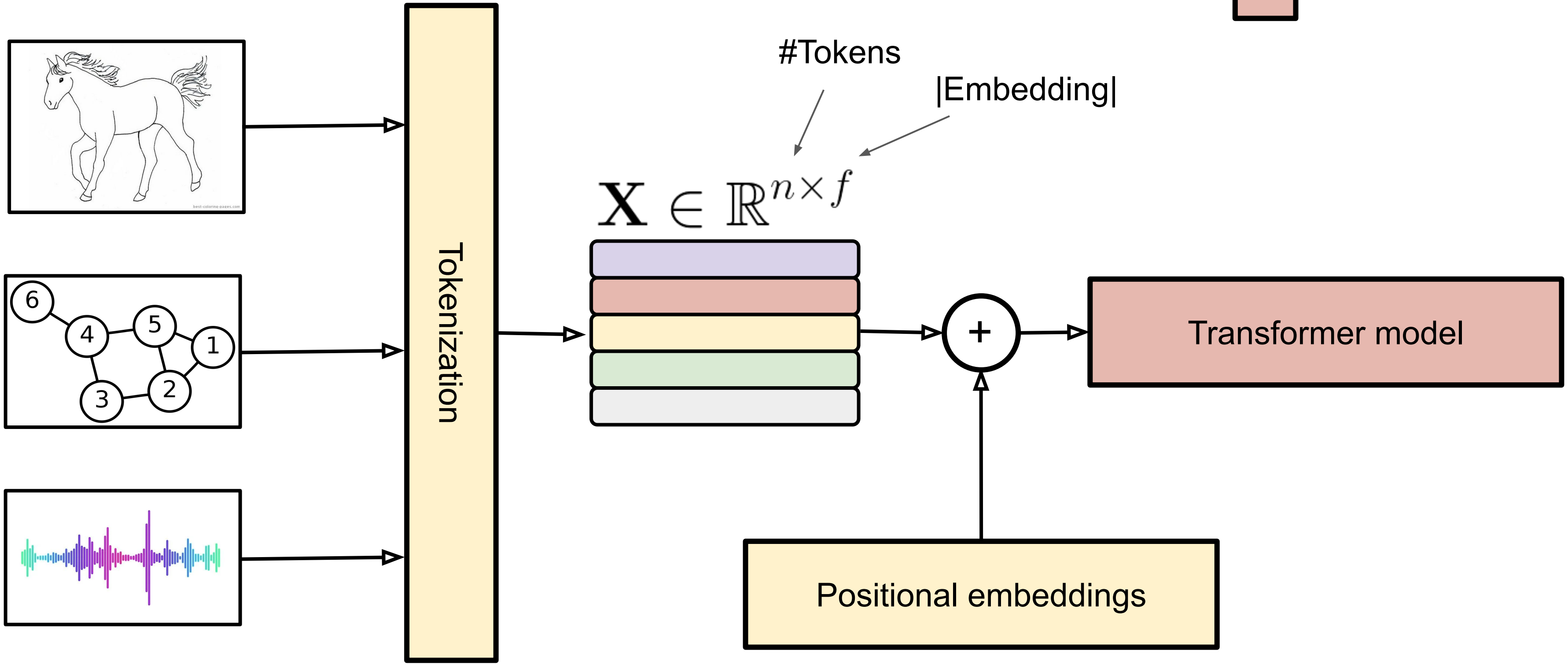


Figure 2. **Left/Center**: Representation quality, measured as ImageNet finetune and linear 10-shot error rate, as a function of total training compute. A saturating power-law approximates the Pareto frontier fairly accurately. Note that smaller models (blue shading), or models trained on fewer images (smaller markers), saturate and fall off the frontier when trained for longer. **Top right**: Representation quality when bottlenecked by model size. For each model size, a large dataset and amount of compute is used, so model capacity is the main bottleneck. Faintly-shaded markers depict sub-optimal runs of each model. **Bottom Right**: Representation quality by datasets size. For each dataset size, the model with an optimal size and amount of compute is highlighted, so dataset size is the main bottleneck.

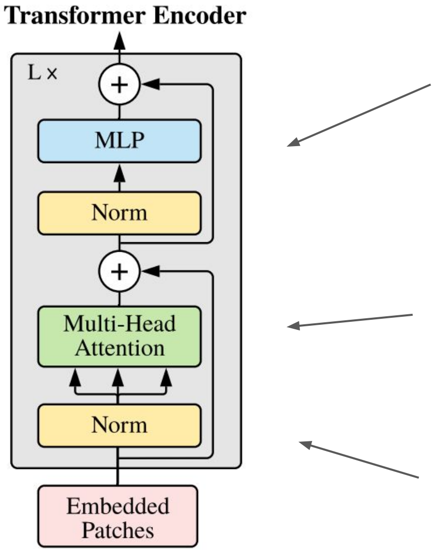
Transformers at a glance

🟨 Input-dependent

🟥 Input-independent



Zooming in



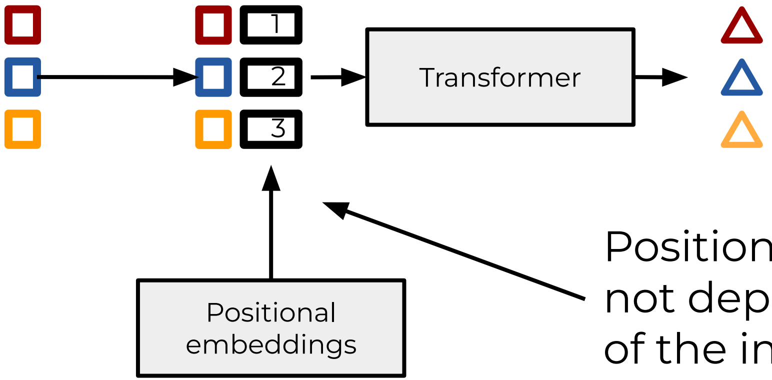
A small neural network (e.g., 2 layers) applied to each token independently.
Fast, average number of parameters.

Main component, allowing to combine information *across* different tokens.
Quadratic in the number of tokens!

Normalization (typically layer normalization): helps in stabilizing mean and variance of the embeddings.
Very fast, small number of parameters.

Positional embeddings

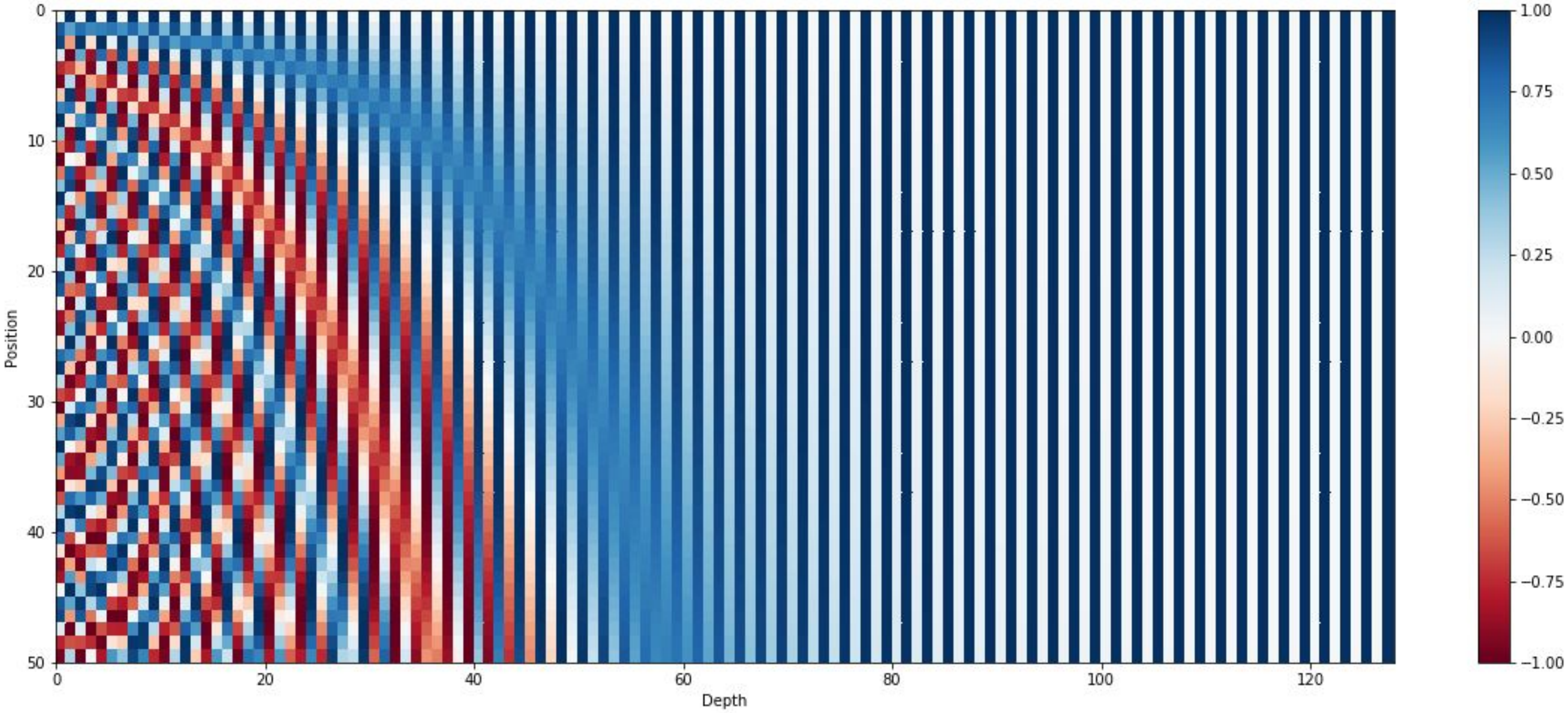
This vector should encode information about the position of the corresponding token.



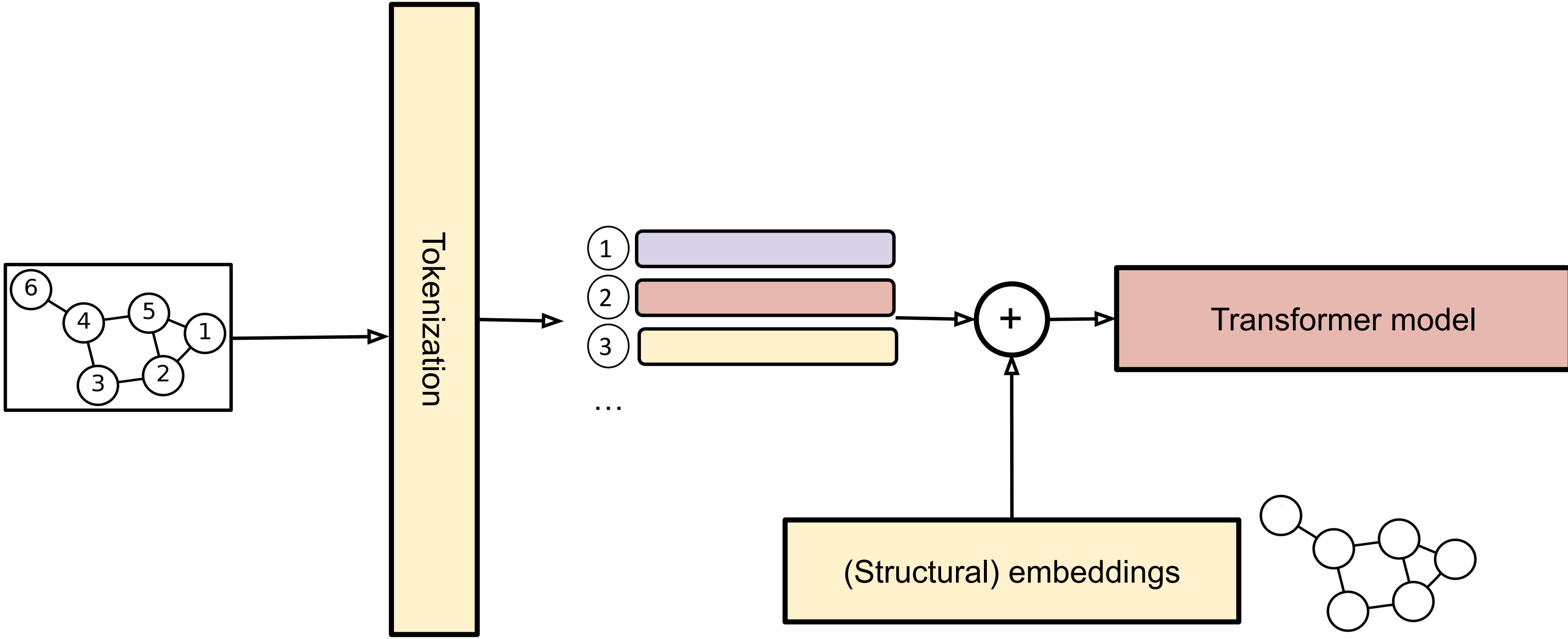
Positional embeddings do not depend on the *content* of the image or input.

Sinusoidal embeddings

Transformer Architecture: The Positional Encoding - Amirhossein Kazemnejad's Blog



Example of (graph) transformers



Taxonomy of graph transformers

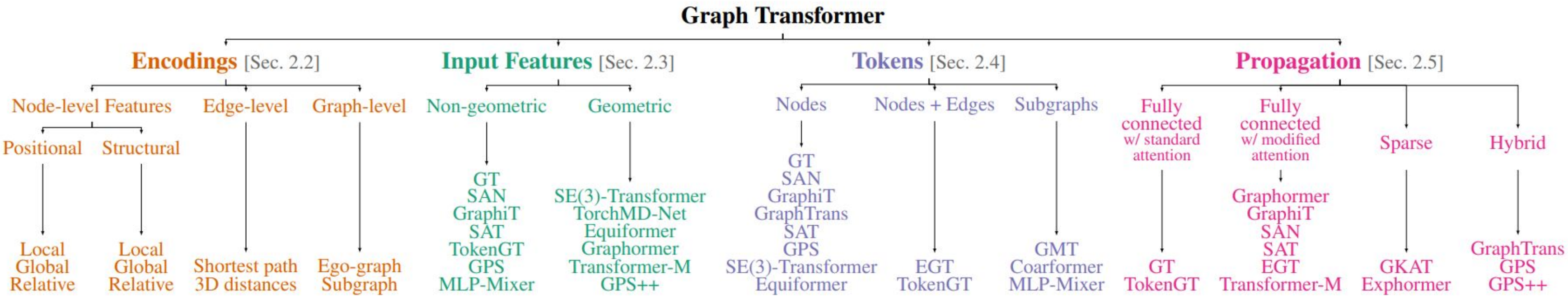

Graph Transformer
Encodings [Sec. 2.2]
Input Features [Sec. 2.3]
Tokens [Sec. 2.4]
Propagation [Sec. 2.5]
Node-level Features
Edge-level
Graph-level
Non-geometric
Geometric
Nodes
Nodes + Edges
Subgraphs
Fully connected w/ standard attention
Fully connected w/ modified attention
Sparse
Hybrid
Positional
Structural
Local Global Relative
Local Global Relative
Shortest path 3D distances
Ego-graph Subgraph
GT SAN GraphiT SAT TokenGT GPS MLP-Mixer
SE(3)-Transformer TorchMD-Net Equiformer Graphormer Transformer-M GPS++
GT SAN GraphiT GraphTrans SAT GPS SE(3)-Transformer Equiformer
EGT TokenGT
GMT Coarformer MLP-Mixer
GT TokenGT
Graphormer GraphiT SAN SAT EGT Transformer-M
GKAT Exphormer
GraphTrans GPS GPS++

Figure 1: Categorization of graph transformers along four main categories with representative architectures.

[2302.04181] Attending to Graph Transformers

Graph transformers

✔ Each MHA operation has a global receptive field over the graph (better for long-range interactions).

❌ Much harder to design structural embeddings to encode the graph connectivity (e.g., random walks, Laplacian embeddings).

❌ Attention is quadratic in the number of tokens.

No clear scaling law for graphs / unclear expressiveness compared to GNNs.

Beyond graphs

Hypergraphs and **latent inference**

Scaling to higher-order structures

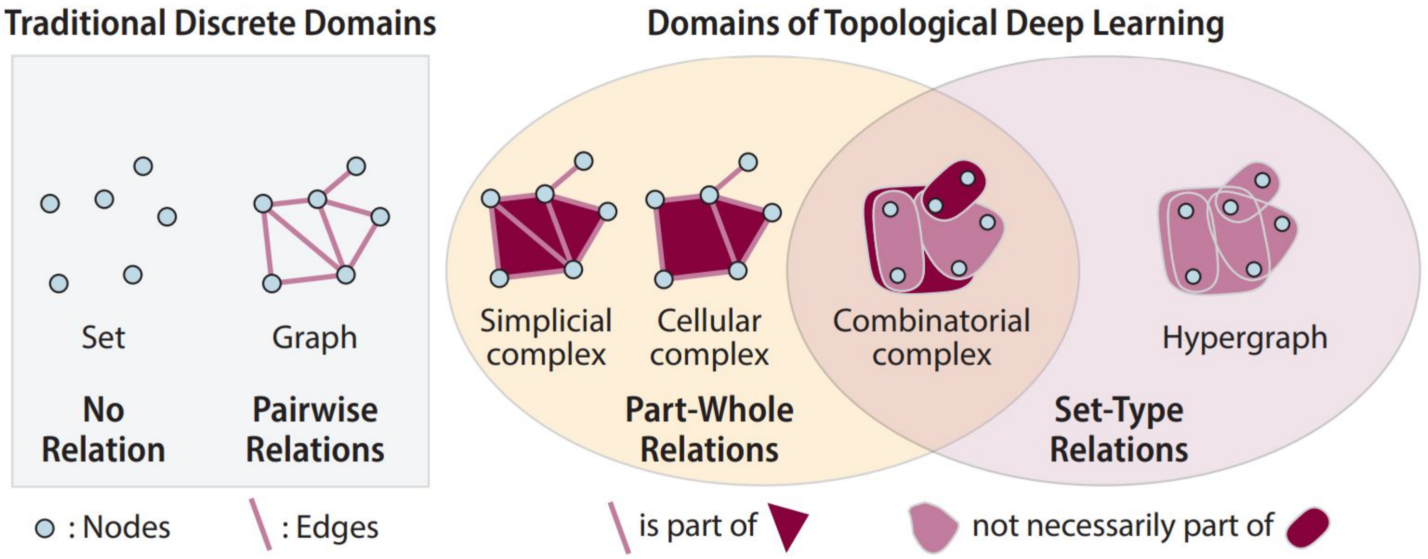
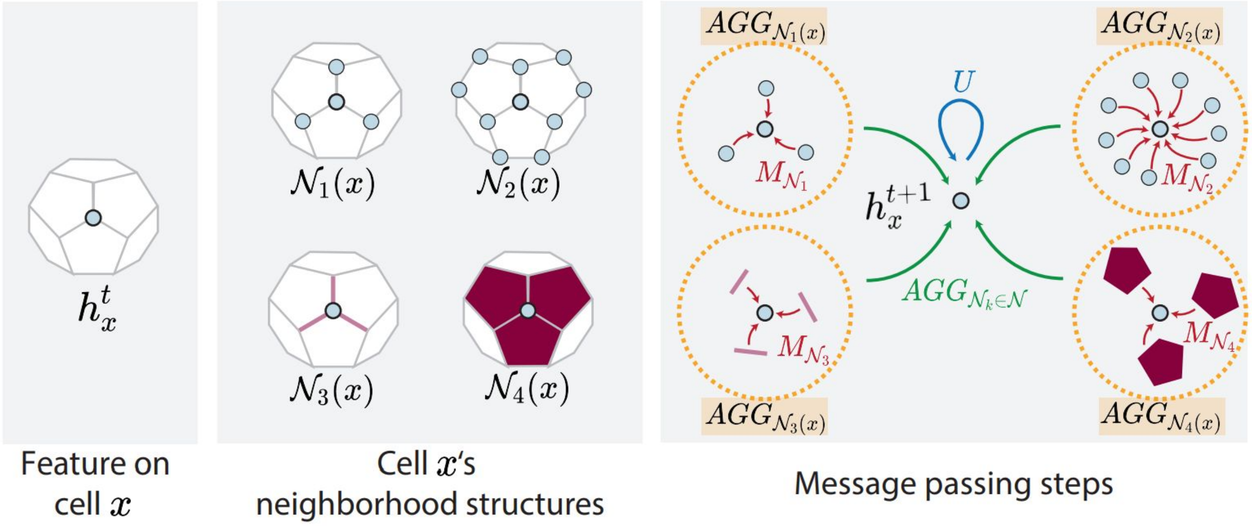


Figure 2: Domains: Nodes in blue, (hyper)edges in pink, and faces in dark red. Inspired by Hajij et al. (2022a).

[2304.10031] Architectures of Topological Deep Learning: A Survey on Topological Neural Networks

Topological message-passing



Feature on cell x

Cell x's neighborhood structures

Message passing steps

Figure 8: Message passing steps: 1: Message (red), 2: Within-neighborhood aggregation (orange), 3: Between-neighborhood aggregation (green), 4: Update (blue). The scheme updates a feature $\mathbf{h}_x^{t,(r)}$ on a r-cell x at layer t (left column) into a new feature $\mathbf{h}_x^{t+1,(r)}$ on that same cell at the next layer $t+1$ (right column). Here, the scheme uses four neighborhood structures $\mathcal{N}_k$ for $k \in \{1, 2, 3, 4\}$ (middle column). Inspired by (Hajij et al., 2023).

[2304.10031] Architectures of Topological Deep Learning: A Survey on Topological Neural Networks

Topological lifting

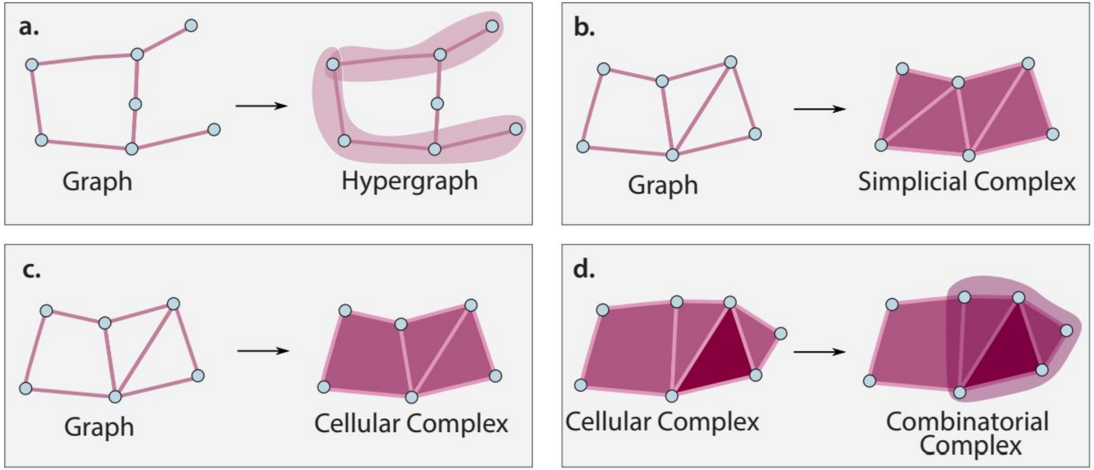
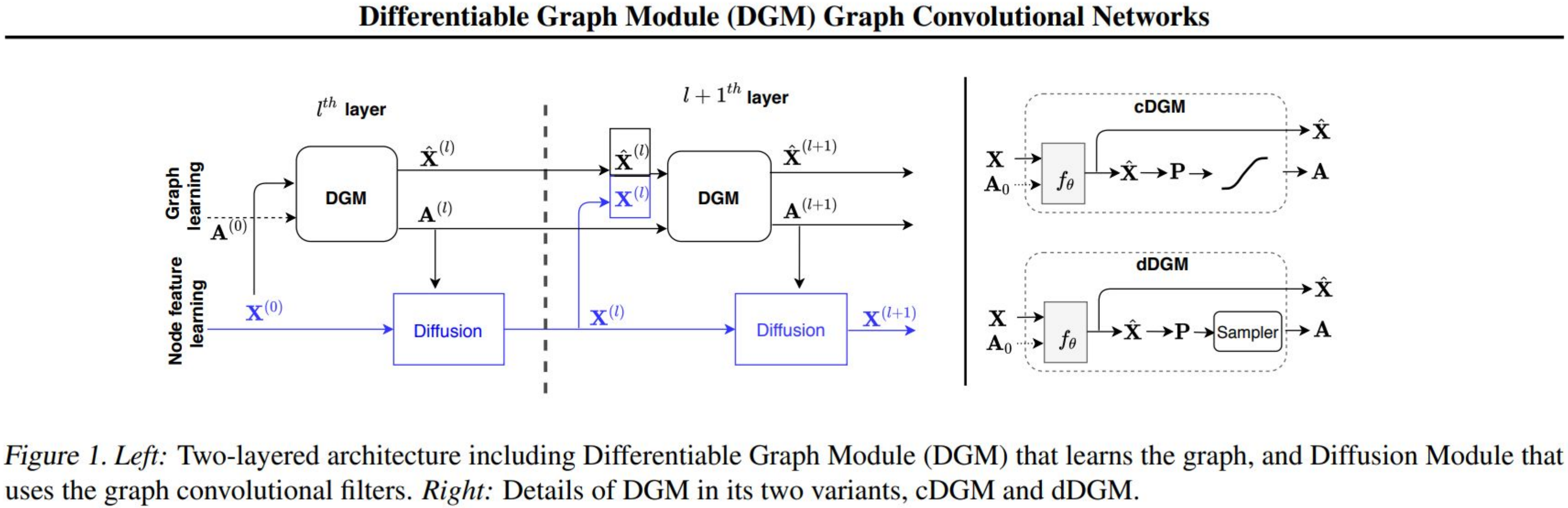


Figure 4: Lifting Topological Domains. (a) A graph is "lifted" to a hypergraph by adding hyperedges that connect groups of nodes. (b) In the process of lifting a graph to a simplicial complex, a pairwise edge must be added in order to form triangular faces. (c) A graph can be converted to a cellular complex by adding faces of any shape. (d) Hyperedges can be added to a cellular complex to lift the structure to a combinatorial complex. Figure adapted from Hajij et al. (2023).

[2304.10031] Architectures of Topological Deep Learning: A Survey on Topological Neural Networks

Learning the latent connectivity



Differentiable Graph Module (DGM) Graph Convolutional Networks

Figure 1. Left: Two-layered architecture including Differentiable Graph Module (DGM) that learns the graph, and Diffusion Module that uses the graph convolutional filters. *Right:* Details of DGM in its two variants, cDGM and dDGM.

Kazi, A., Cosmo, L., Ahmadi, S.A., Navab, N. and Bronstein, M., 2022. **Differentiable graph module (dgm) for graph convolutional networks**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Learning the *topological* connectivity

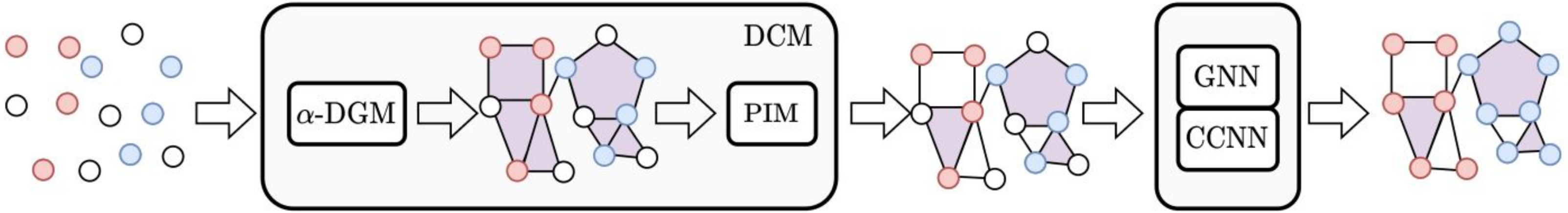


Figure 1: The proposed two-step procedure for Latent Topology Inference (LTI) via regular cell complexes. The Differentiable Cell Complex Module (DCM) is a function that first learns a graph describing the pairwise interactions among data points via the α-Differentiable Graph Module (α-DGM), and then it leverages the graph as the 1-skeleton of a regular cell complex whose 2-cells (polygons), describing multi-way interactions among data points, are learned via the Polygon Inference Module (PIM). The inferred topology is then used in two message passing networks, at node (Graph Neural Network, GNN) and edge (Cell Complex Neural Network, CCNN) levels to solve the downstream task. The whole architecture is trained in a end-to-end fashion.

Battiloro, C., Spinelli, I., Telyatnikov, L., Bronstein, M., Scardapane, S., & Di Lorenzo, P. (2023). **From Latent Graph to Latent Topology Inference: Differentiable Cell Complex Module**. *arXiv preprint arXiv:2305.16174*.

Introduction

A primer on **explainability**

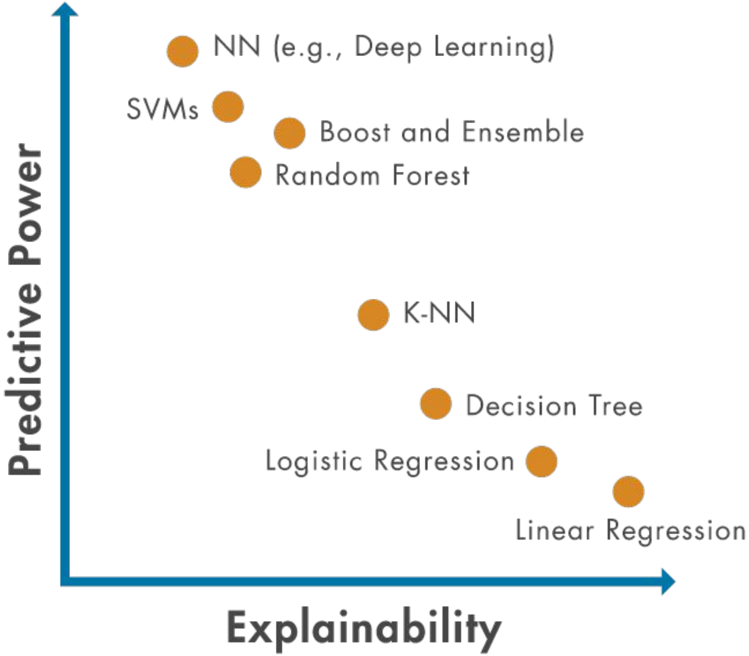
What is explainable AI (XAI)?

1. Broad field concerning the development of tools to increase **trust** and **understanding** of a model's predictions.

2. *Common folk dichotomy*: intrinsically interpretable models (e.g., linear regression, decision trees) are orthogonal to models with strong representational power (e.g., deep networks).

Ras, G., Xie, N., van Gerven, M. and Doran, D., 2022. Explainable Deep Learning: A Field Guide for the Uninitiated. *Journal of Artificial Intelligence Research*, *73*, pp.329-397.

Explainability vs. accuracy?



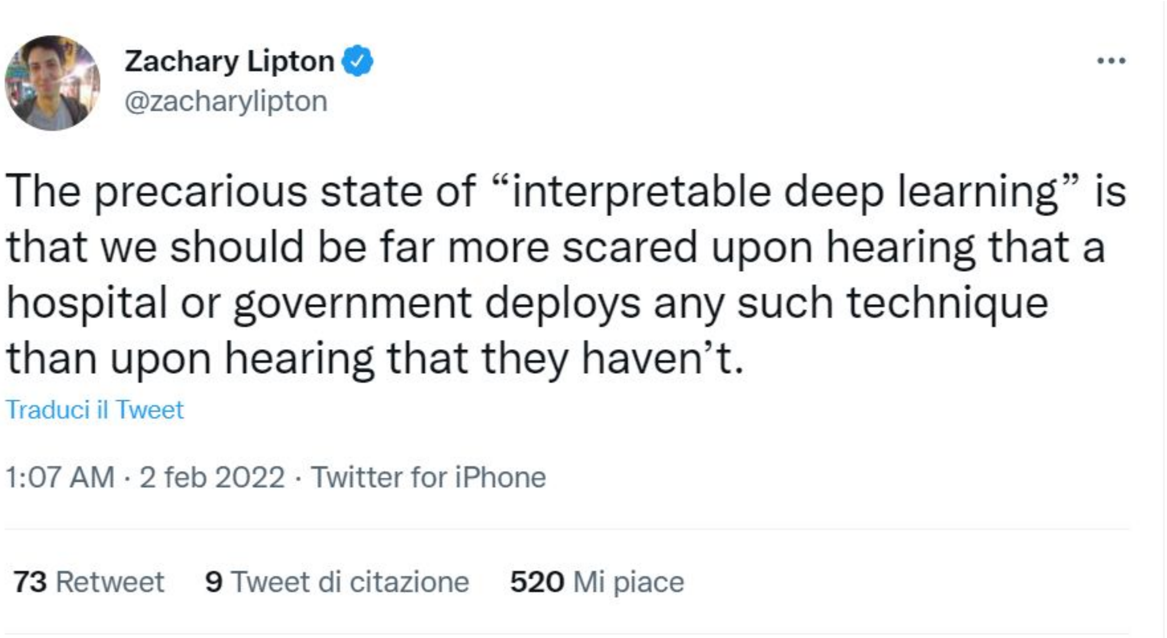
Interpretability - MATLAB & Simulink

Who is explainability for?

1. Most XAI methods are targeted towards **practitioners** of the same methods (i.e., they are akin to debugging tools).

2. It is much harder to target XAI tools towards **end-users** (e.g., clinical staff). Different methods may disagree on the "explanation", they may not be accurate, and they lack principled **evaluation metrics**.

Krishna, S., Han, T., Gu, A., Pombra, J., Jabbari, S., Wu, S. and Lakkaraju, H., 2022. The Disagreement Problem in Explainable Machine Learning: A Practitioner's Perspective. *arXiv preprint arXiv:2202.01602*.



Zachary Lipton
@zacharylipton

The precarious state of “interpretable deep learning” is that we should be far more scared upon hearing that a hospital or government deploys any such technique than upon hearing that they haven’t.

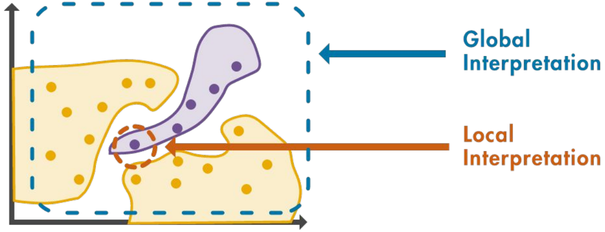
Traduci il Tweet

1:07 AM · 2 feb 2022 · Twitter for iPhone

73 Retweet **9** Tweet di citazione **520** Mi piace

Arora, S., Pruthi, D., Sadeh, N., Cohen, W.W., Lipton, Z.C. and Neubig, G., 2021. Explain, Edit, and Understand: Rethinking User Study Design for Evaluating Model Explanations. *AAAI 2022*.

XAI categorization



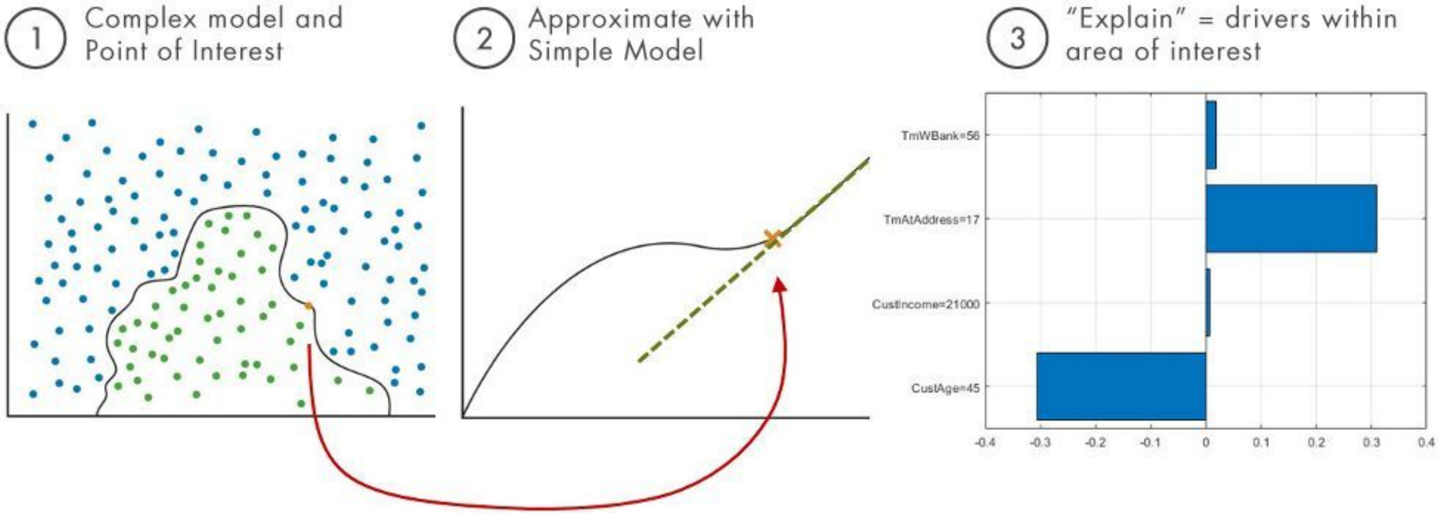
1. XAI tools can be categorized depending on whether they provide **global** or **local** explanations.

2. Some methods are **model-agnostic** (they only need the outputs of the models), other are **model-specific**.

3. Finally, methods can be categorized depending on what type of information they provide in output.

Molnar, C., 2020. Interpretable machine learning. Independently published.

Global vs. local explanations


1
Complex model and Point of Interest
2
Approximate with Simple Model
3
"Explain" = drivers within area of interest
TmWBank=56
TmAtAddress=17
CustIncome=21000
CustAge=45
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4

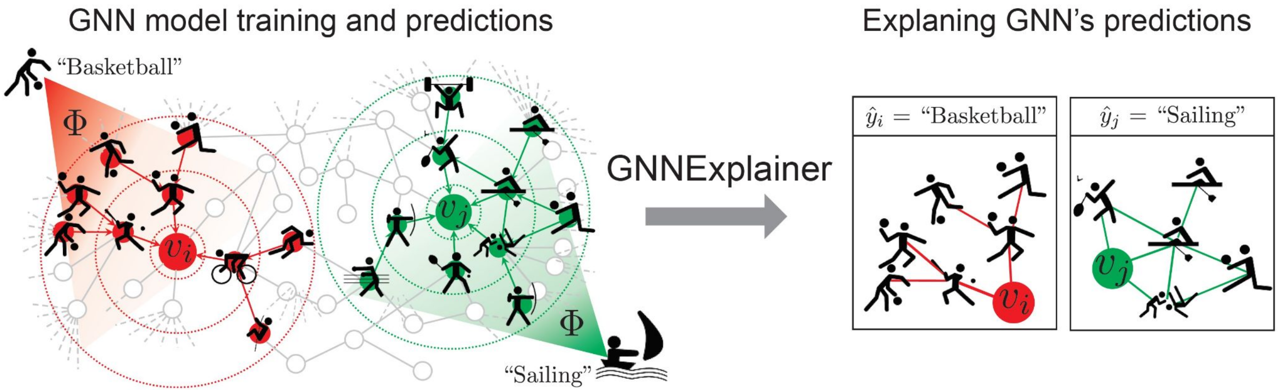
Interpretability - MATLAB & Simulink

Introduction

Local explanations for graph NNs

Explaining a graph NN

Consider a trained graph NN. In the majority of works, a **local explanation** is a small subgraph (and subset of features) which "explain" the prediction.


GNN model training and predictions
"Basketball"
Φ
v_i
v_j
Φ
"Sailing"
GNNExplainer
Explaning GNN's predictions
$\hat{y}_i$ = "Basketball"
$\hat{y}_j$ = "Sailing"
v_i
v_j

Masked predictions

$$f(\mathbf{M}_X \odot \mathbf{X}, \mathbf{M}_A \odot \mathbf{A}) \approx f(\mathbf{X}, \mathbf{A})$$

Binary masks

Original prediction

A “good” explanation should have specific properties (e.g., smallest possible masks).

Evaluating explanations

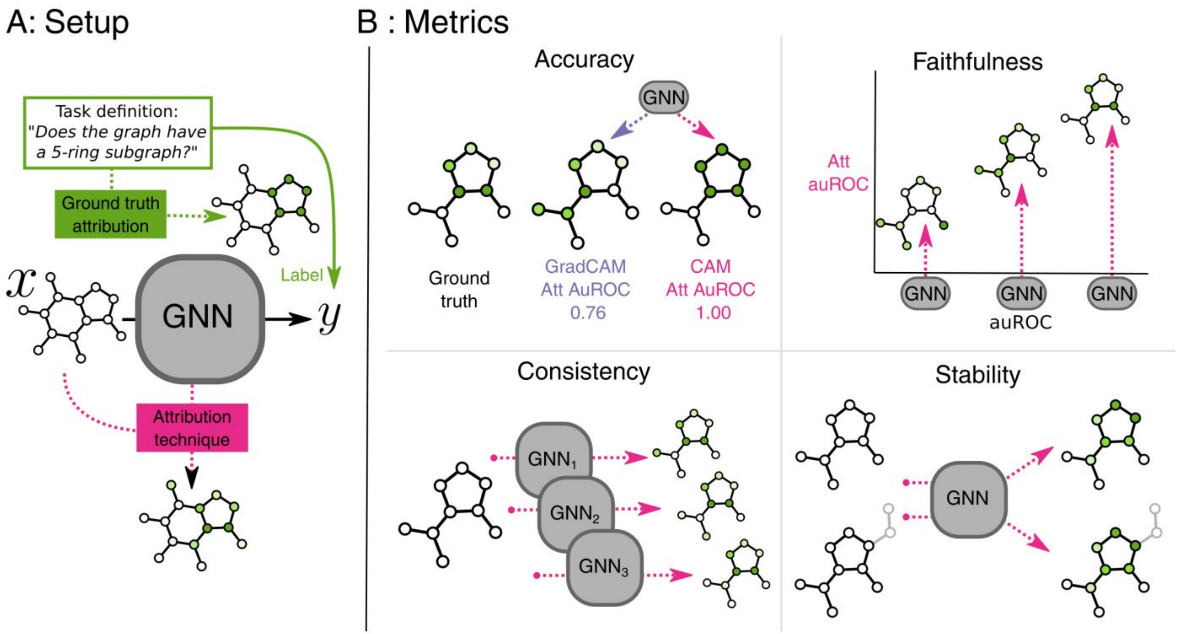


Figure 1: Schematic of attribution task setup and attribution metrics. **A**. We create classification and regression tasks for which we have a computable ground-truth. We train GNN models on these labels, and calculate attributions using the graph inputs and attribution methods we adapt to graphs. **B**. We quantify attribution performance with four metrics. *Accuracy* measures how well an attribution matches ground-truth. *Consistency* measures how accuracy varies across different hyperparameters of a model. *Faithfulness* measures how well the performance of an attribution method matches model performance. *Stability* measures how attributions change when the input is perturbed.

Sanchez-Lengeling, B., Wei, J., Lee, B., Reif, E., Wang, P., Qian, W., McCloskey, K., Colwell, L. and Wiltschko, A., 2020. Evaluating attribution for graph neural networks. *Advances in neural information processing systems*, *33*, pp.5898-5910.

“Classic” XAI methods

Attribution techniques

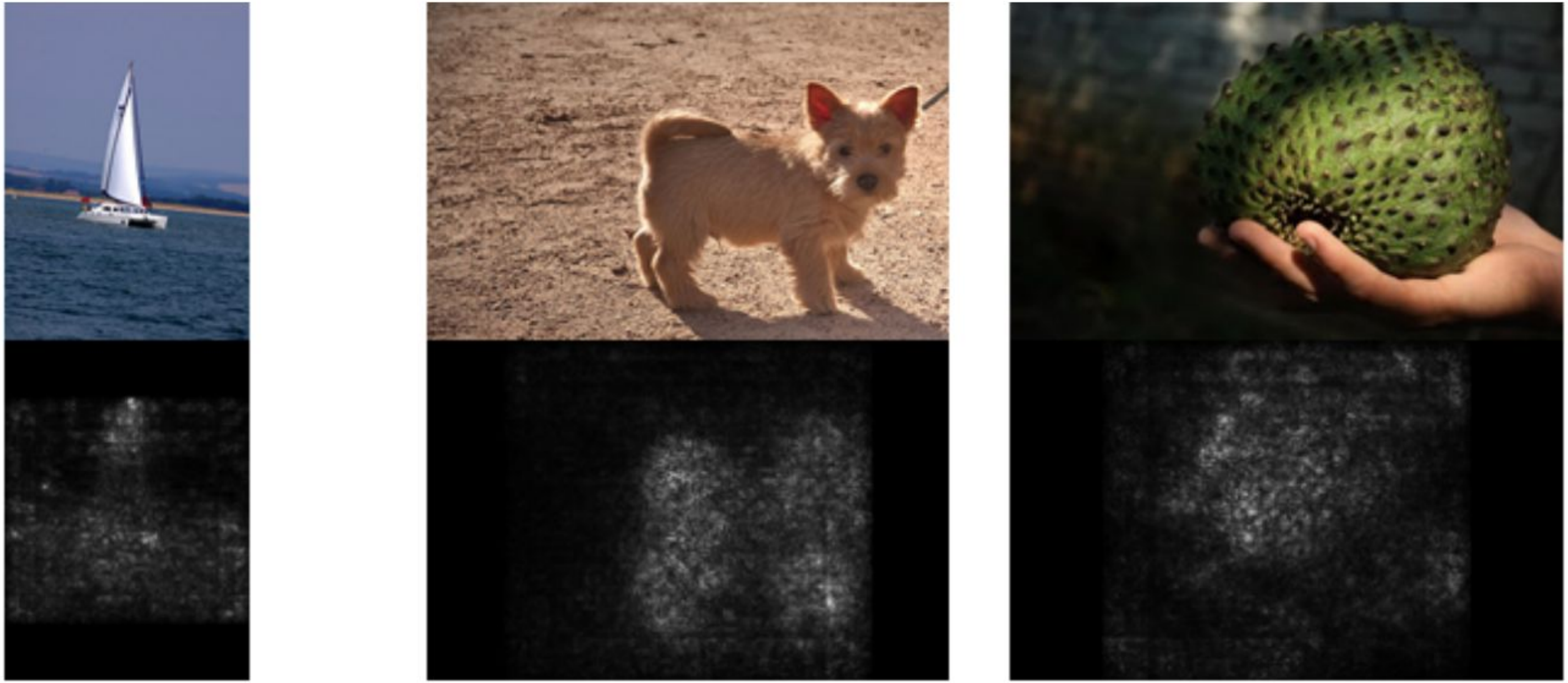
Saliency maps

A **saliency map** is an object of the same dimensionality as the input, providing information about which features were most important for a given prediction.

Formally (*i* is the index of the class of interest):

$$\text{Saliency map} = \max_{\text{channels}} \left| \frac{\partial f_i(x)}{\partial x} \right|$$

Simonyan, K., Vedaldi, A. and Zisserman, A., 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.



Saliency maps for graphs

A very similar procedure can be done for a graph NN, to obtain graph saliency maps:

$$\mathbf{M}_A = \frac{\partial f(\mathbf{X}, \mathbf{A})}{\partial \mathbf{A}}$$

(We focus mostly on edge saliency maps, as they are easier to visualize.)

Baldassarre, F. and Azizpour, H., 2019. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686*.

Example

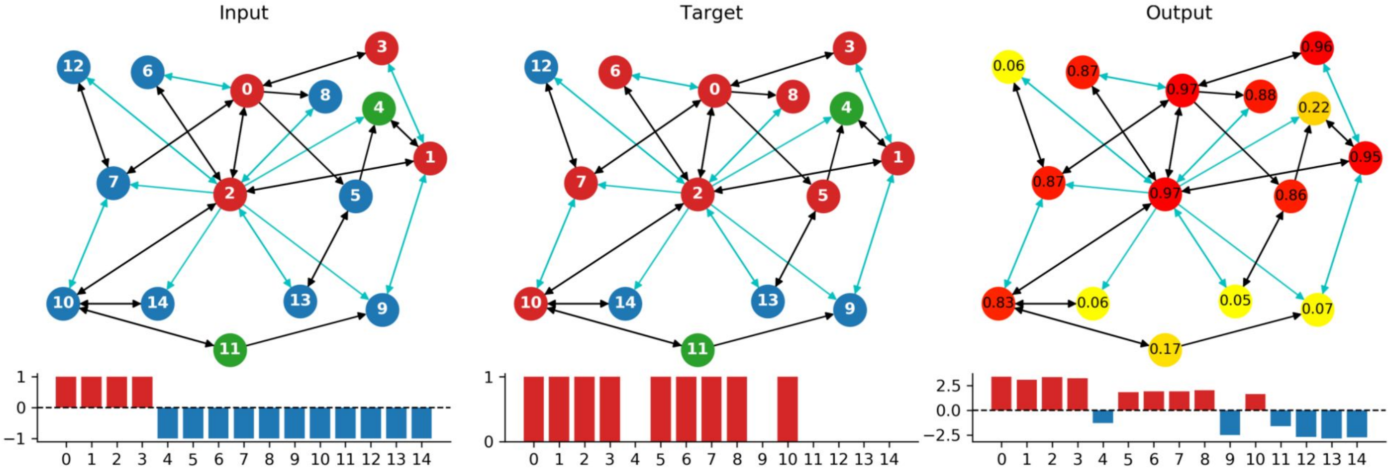
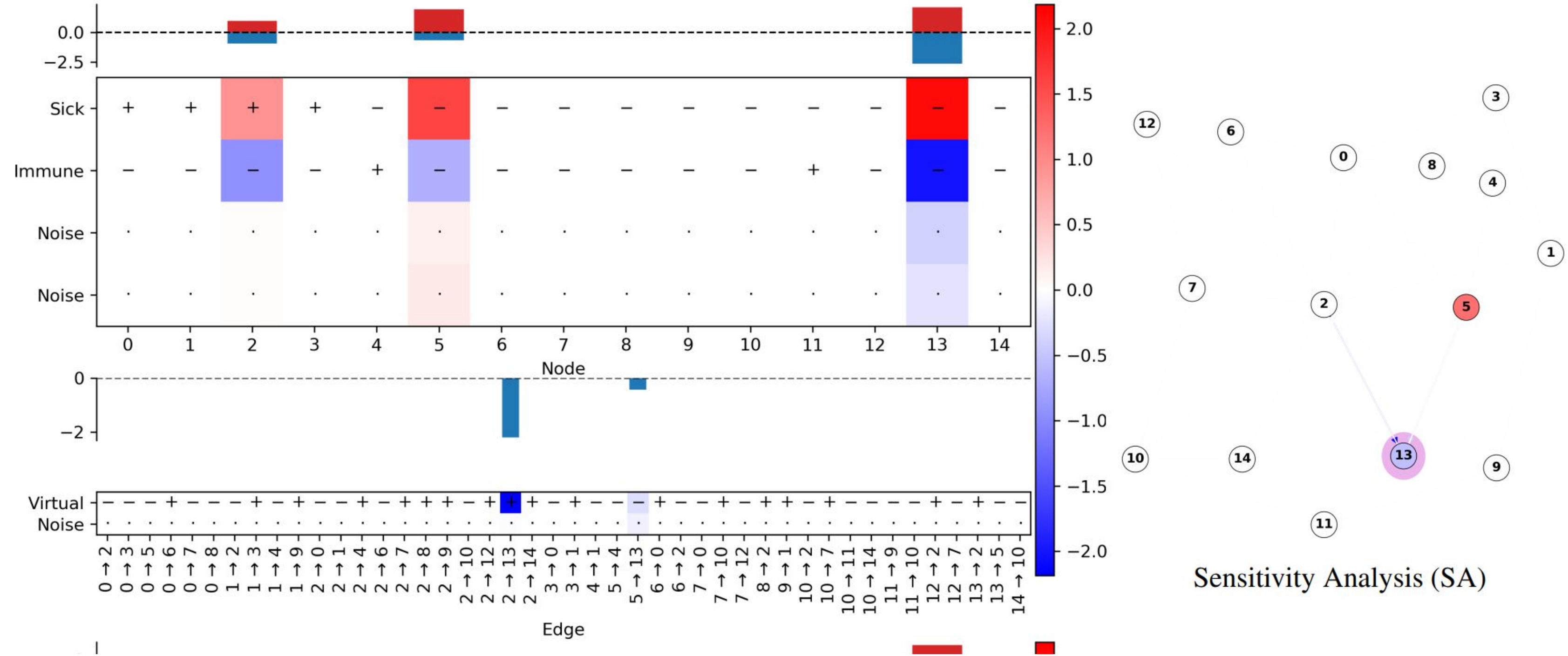


Figure 8: **Graph 1: nodes 0, 1, 2 and 3 are initially sick; nodes 4 and 11 are immune; the others are healthy.** After one propagation step, the infection reaches nodes 5, 6, 7 and 10. The network predicts the correct label for every node of the graph, following the spread of the infection along non-virtual edges to non-immune nodes. The figures that follow are a visualization of the explanations produced for nodes: 10, 13, 4

Baldassarre, F. and Azizpour, H., 2019. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686*.

Example



Sensitivity Analysis (SA)

Baldassarre, F. and Azizpour, H., 2019. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686*.

Limits of saliency maps

Simple saliency maps have issues that balances their simplicity:

1. They are highly **unstable** wrt small changes in the input.
2. They are not well **localized**.
3. They have no formal guarantees.

In particular, they do not respect a property called **sensitivity**: if two inputs differ for a single element but have different predictions, a saliency map is not guaranteed to highlight that pixel.

Integrated gradients

In the CV field, **integrated gradients** are a powerful alternative to standard saliency maps.

They recover sensitivity by integrating the gradients along a path moving from the empty image to the current one.

We can do something similar by considering an empty adjacency matrix A0:

$$\mathbf{M}_A = (\mathbf{A} - \mathbf{A}_0) \int_{\alpha \in [0,1]} \frac{\partial f(\mathbf{X}, \alpha \mathbf{A} + (1 - \alpha)\mathbf{A_0})}{\partial \mathbf{A}}$$

Sundararajan, M., Taly, A. and Yan, Q., 2017, July. Axiomatic attribution for deep networks. In *International conference on machine learning* (pp. 3319-3328). PMLR.

Some benchmarks

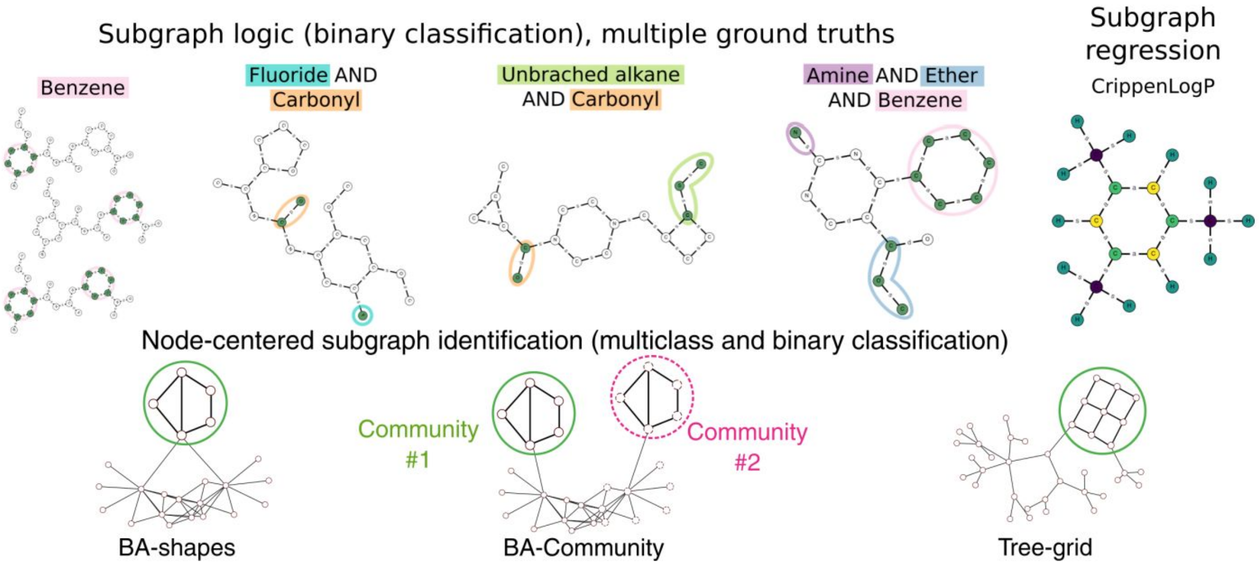
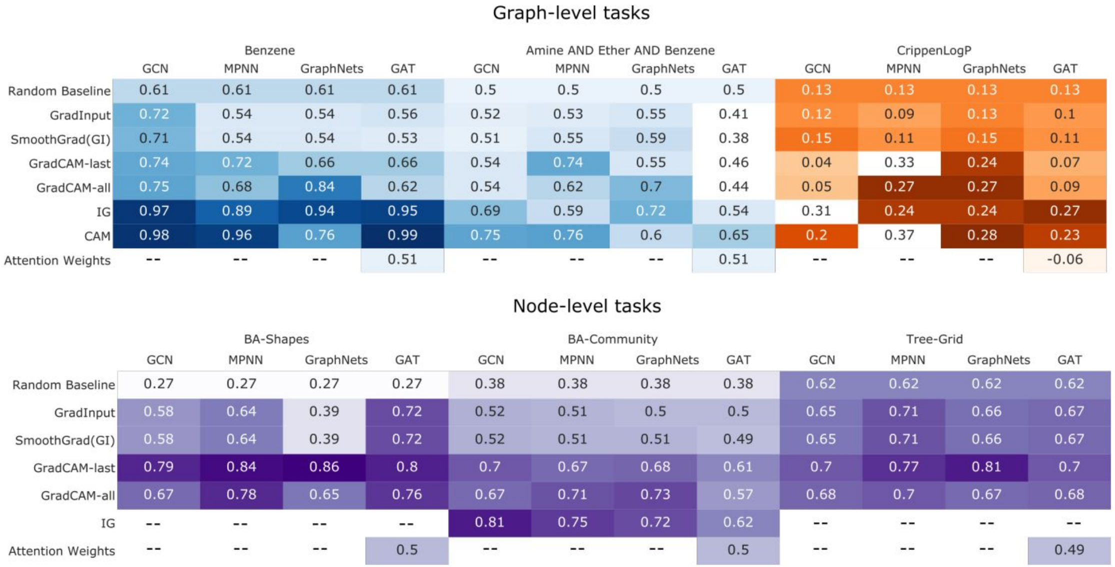

Subgraph logic (binary classification), multiple ground truths
Subgraph regression
CrippenLogP
Benzene
Fluoride AND Carbonyl
Unbrached alkane AND Carbonyl
Amine AND Ether AND Benzene
Node-centered subgraph identification (multiclass and binary classification)
Community #1
Community #2
BA-shapes
BA-Community
Tree-grid

Figure 2: Example ground truth attributions for each task. The first four graph-classification tasks require a model to identify all nodes (green) in one or more subgraphs (colored lasso) in molecular graphs. Each graph may have multiple positive ground-truths, shown in the *Benzene* task. Ground truth attributions for the *CrippenLogP* regression task take on continuous values. Lower row has node-classification tasks. Relevant subgraphs are circled. Only one neighborhood of the graph is shown.

Sanchez-Lengeling, B., Wei, J., Lee, B., Reif, E., Wang, P., Qian, W., McCloskey, K., Colwell, L. and Wiltschko, A., 2020. Evaluating attribution for graph neural networks. *Advances in neural information processing systems*, *33*, pp.5898-5910.

Some benchmarks (2)



Graph-level tasks

	Benzene				Amine AND Ether AND Benzene				CrippenLogP			
	GCN	MPNN	GraphNets	GAT	GCN	MPNN	GraphNets	GAT	GCN	MPNN	GraphNets	GAT
Random Baseline	0.61	0.61	0.61	0.61	0.5	0.5	0.5	0.5	0.13	0.13	0.13	0.13
GradInput	0.72	0.54	0.54	0.56	0.52	0.53	0.55	0.41	0.12	0.09	0.13	0.1
SmoothGrad(GI)	0.71	0.54	0.54	0.53	0.51	0.55	0.59	0.38	0.15	0.11	0.15	0.11
GradCAM-last	0.74	0.72	0.66	0.66	0.54	0.74	0.55	0.46	0.04	0.33	0.24	0.07
GradCAM-all	0.75	0.68	0.84	0.62	0.54	0.62	0.7	0.44	0.05	0.27	0.27	0.09
IG	0.97	0.89	0.94	0.95	0.69	0.59	0.72	0.54	0.31	0.24	0.24	0.27
CAM	0.98	0.96	0.76	0.99	0.75	0.76	0.6	0.65	0.2	0.37	0.28	0.23
Attention Weights	--	--	--	0.51	--	--	--	0.51	--	--	--	-0.06

Node-level tasks

	BA-Shapes				BA-Community				Tree-Grid			
	GCN	MPNN	GraphNets	GAT	GCN	MPNN	GraphNets	GAT	GCN	MPNN	GraphNets	GAT
Random Baseline	0.27	0.27	0.27	0.27	0.38	0.38	0.38	0.38	0.62	0.62	0.62	0.62
GradInput	0.58	0.64	0.39	0.72	0.52	0.51	0.5	0.5	0.65	0.71	0.66	0.67
SmoothGrad(GI)	0.58	0.64	0.39	0.72	0.52	0.51	0.51	0.49	0.65	0.71	0.66	0.67
GradCAM-last	0.79	0.84	0.86	0.8	0.7	0.67	0.68	0.61	0.7	0.77	0.81	0.7
GradCAM-all	0.67	0.78	0.65	0.76	0.67	0.71	0.73	0.57	0.68	0.7	0.67	0.68
IG	--	--	--	--	0.81	0.75	0.72	0.62	--	--	--	--
Attention Weights	--	--	--	0.5	--	--	--	0.5	--	--	--	0.49

Figure 3: Attribution method accuracy across tasks and model architectures. Colors are used to distinguish two metric types — attribution AUROC for attribution on classification tasks, and attribution Kendall's tau on the regression task. CAM and IG perform consistently well across tasks and models. For error bars, please see Figure S1.

Sanchez-Lengeling, B., Wei, J., Lee, B., Reif, E., Wang, P., Qian, W., McCloskey, K., Colwell, L. and Wiltschko, A., 2020. Evaluating attribution for graph neural networks. *Advances in neural information processing systems*, *33*, pp.5898-5910.

Tailored XAI methods

GNNExplainer

GNNExplainer

For graphs, it is fundamental that the resulting subgraph is small, since visualizing and interpreting it is complex.

For this reason, tailored methods have been proposed to force sparsity as the main concern.

GNNExplainer works by optimizing the masks using the following criteria:

1. Keeping the original prediction consistent;
2. Having small masks (l1 regularization);

Ying, R., Bourgeois, D., You, J., Zitnik, M. and Leskovec, J., 2019. Gnn explainer: A tool for post-hoc explanation of graph neural networks. *arXiv preprint arXiv:1903.03894*.

GNNExplainer cost function

$$\mathbf{M}_A = \arg\min_{\mathbf{M}} \boxed{\mathrm{CE}(f(\mathbf{X}, \mathbf{A}), f(\mathbf{X}, \mathbf{M} \odot \mathbf{A}))} + \boxed{\alpha \|\mathbf{M}\|_1} + \boxed{\beta \mathbb{H}\left[\mathbf{M}\right]}$$

Prediction should stay consistent

The mask should be **sparse**

Weights should be as close as possible to 0 or 1

Ying, R., Bourgeois, D., You, J., Zitnik, M. and Leskovec, J., 2019. Gnn explainer: A tool for post-hoc explanation of graph neural networks. *arXiv preprint arXiv:1903.03894*.

Some results

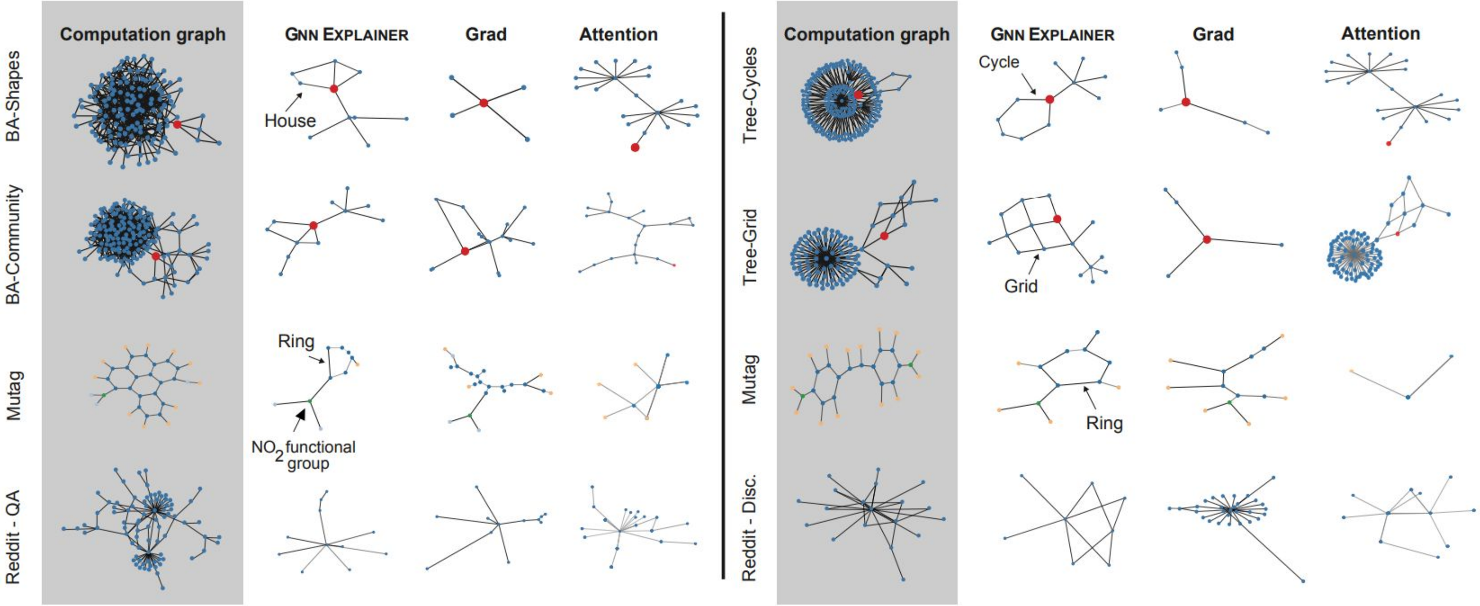


Figure 3: Examples of single-instance important subgraphs. The red node is the explained node.

Ying, R., Bourgeois, D., You, J., Zitnik, M. and Leskovec, J., 2019. Gnn explainer: A tool for post-hoc explanation of graph neural networks. *arXiv preprint arXiv:1903.03894*.

PGExplainer

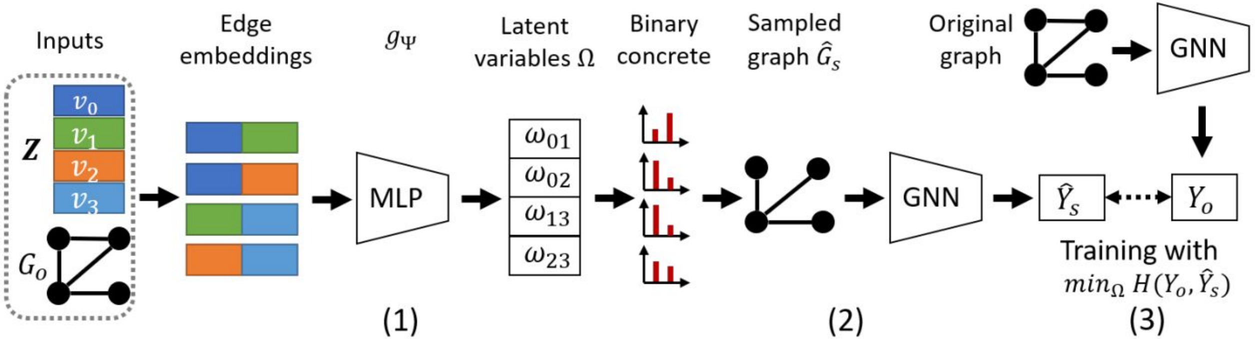


Figure 2: Illustration of PGExplainer for explaining GNNs on graph classification. (1) The left part demonstrates the explanation network. It takes node representations $\mathbf{Z}$ as well as the original graph G_o as inputs to compute Ω, the latent variables in edge distributions. Edge distributions are severed as the explanation. In case that an explanatory subgraph is wanted, we select top-ranked edges according to latent variables Ω. (2) A random graph $\hat{G}_s$ is sampled from edge distributions and then feed to the trained GNN model to get the prediction $\hat{Y}_s$. (3) Parameter Ψ in the explanation network is optimized with cross-entropy between the original prediction Y_o and the updated prediction $\hat{Y}_s$.

Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H. and Zhang, X., 2020. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, *33*, pp.19620-19631.

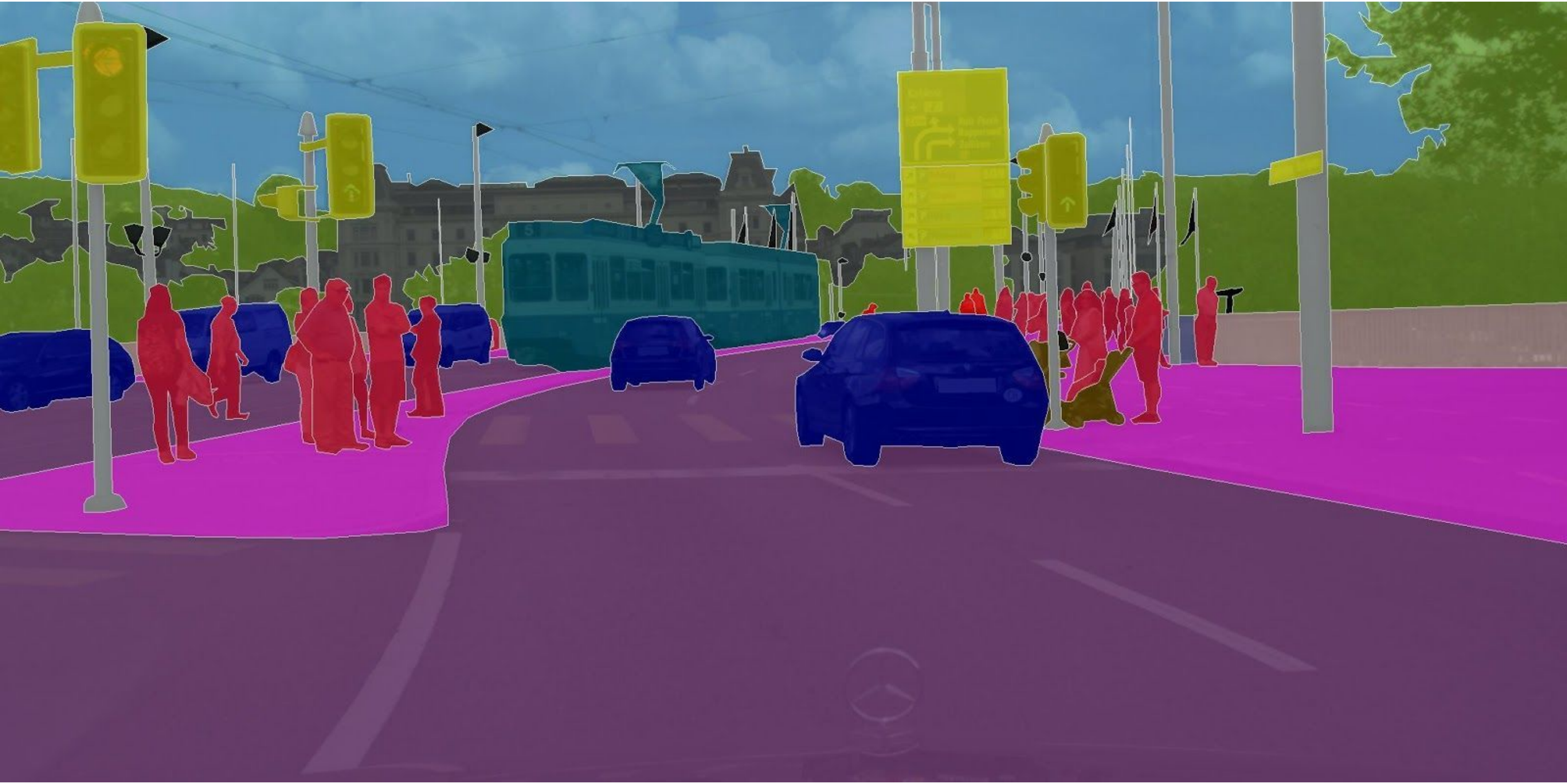
Notebook time!

[Colab Notebooks and Video Tutorials — pytorch_geometric documentation]

https://colab.research.google.com/drive/1nV44NrNqcXC2thU6-zzxnJPnIaIo870m?usp=sharing

Selected topics

Data influence



Limits of saliency maps

1. All the previous methods estimated the influence of single **features** (e.g., edges) on the prediction.
2. An alternative class of methods explore the influence of single **data points** on the prediction, e.g., how much training on a certain graph (or node) has influenced the prediction on a separate graph (or node).
3. This is a more complex scenario, since the influence has to be computed across the entire training run.

Gradient tracing

Consider an idealized training procedure where at iteration *t* we update the parameter vector as:

$$w_{t+1} = w_t - \eta \nabla l(w_t, z_t)$$

The **influence** of point z on point z’ is defined as:

$$\mathrm{TracInIdeal}(z, z') = \sum_{t: z_t = z} l(w_t, z') - l(w_{t+1}, z')$$

Pruthi, G., Liu, F., Kale, S. and Sundararajan, M., 2020. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, *33*, pp.19920-19930.

Gradient tracing

By first-order approximation, it can be shown that:

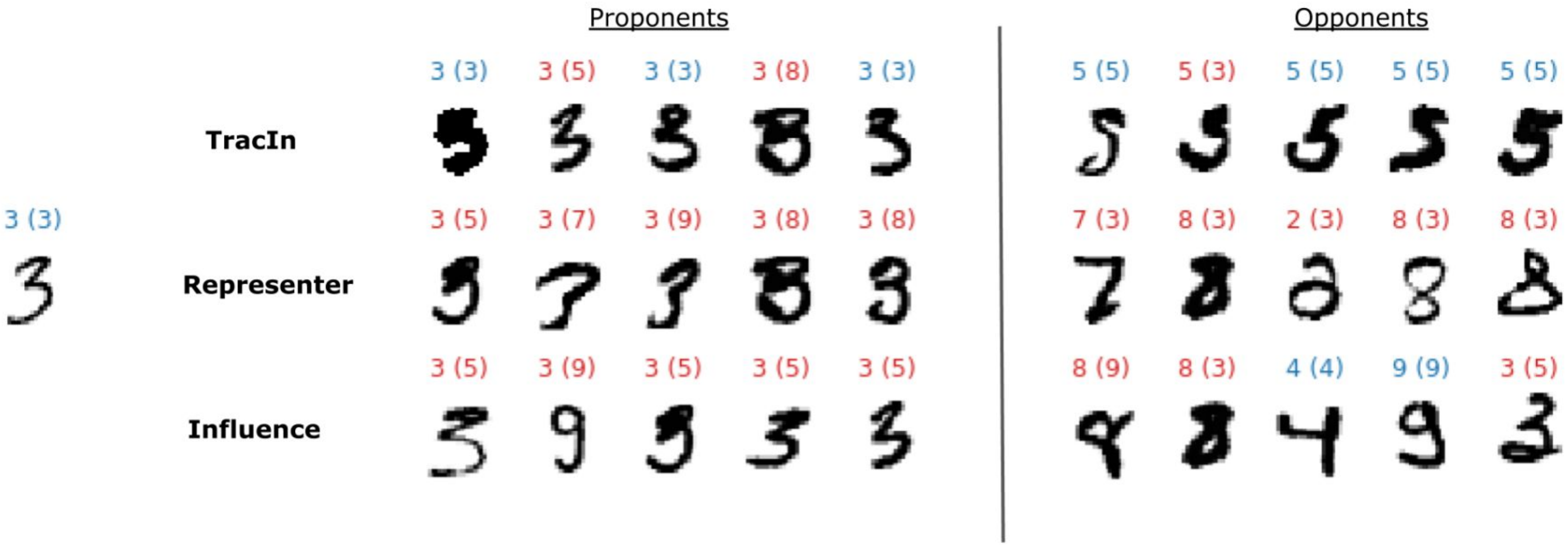
$$\mathrm{TracInIdeal}(z, z') \approx \sum_{t: z_t = z} \eta \nabla l(w_t, z) \cdot \nabla l(w_t, z')$$

This can be approximated by storing *k* checkpoints during training and computing:

$$\mathrm{TracInIdeal}(z, z') \approx \sum_{i=1}^{k} \eta \nabla l(w_i, z) \cdot \nabla l(w_i, z')$$

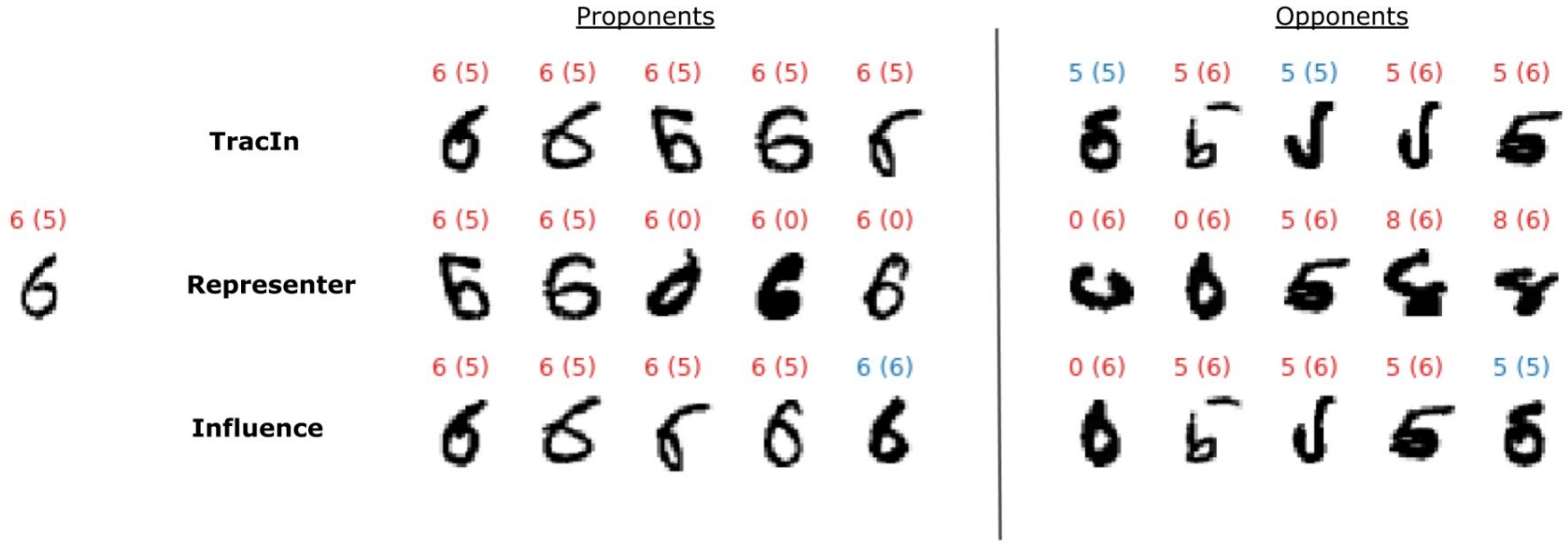
Pruthi, G., Liu, F., Kale, S. and Sundararajan, M., 2020. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, *33*, pp.19920-19930.

Results on CV



(a) Correctly classified 3.

Results on CV



(b) Incorrectly classified 6

Results on CV

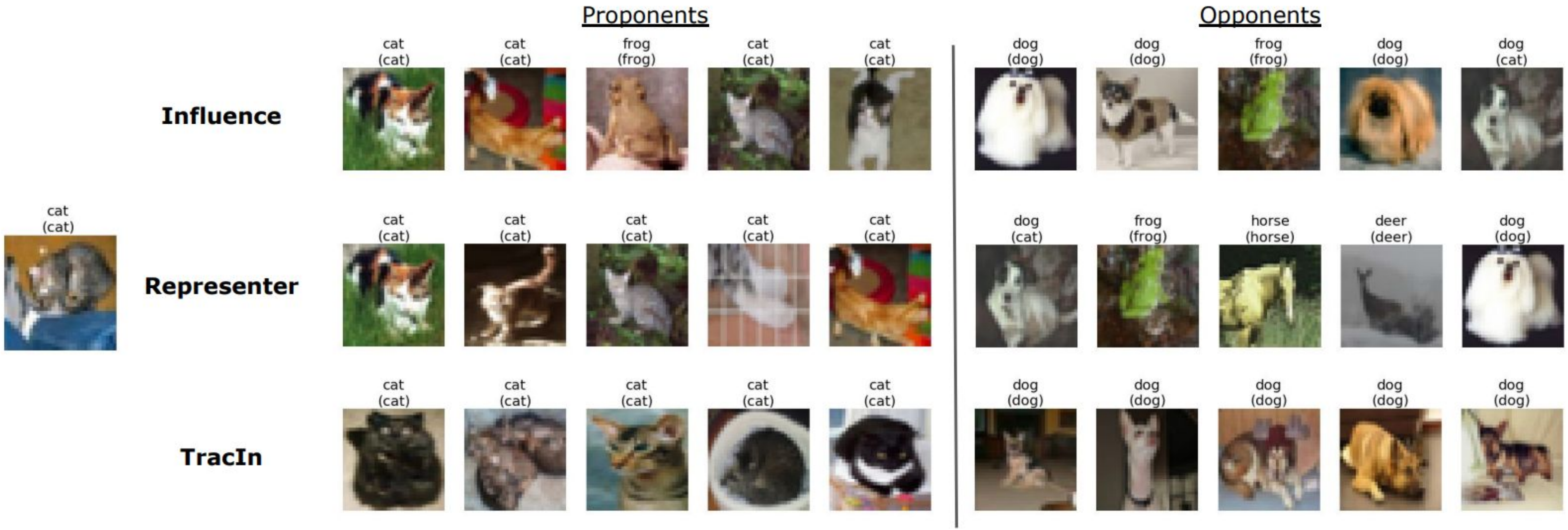


Figure 5: CIFAR-10 results: Proponents and opponents examples of a correctly classified cat for influence functions, representer point, and TracIn. (Predicted class in brackets)

Notebook time!

Colab Notebooks and Video Tutorials — pytorch_geometric documentation

https://colab.research.google.com/drive/1nV44NrNqcXC2thU6-zzxnJPnIalo870m?usp=sharing

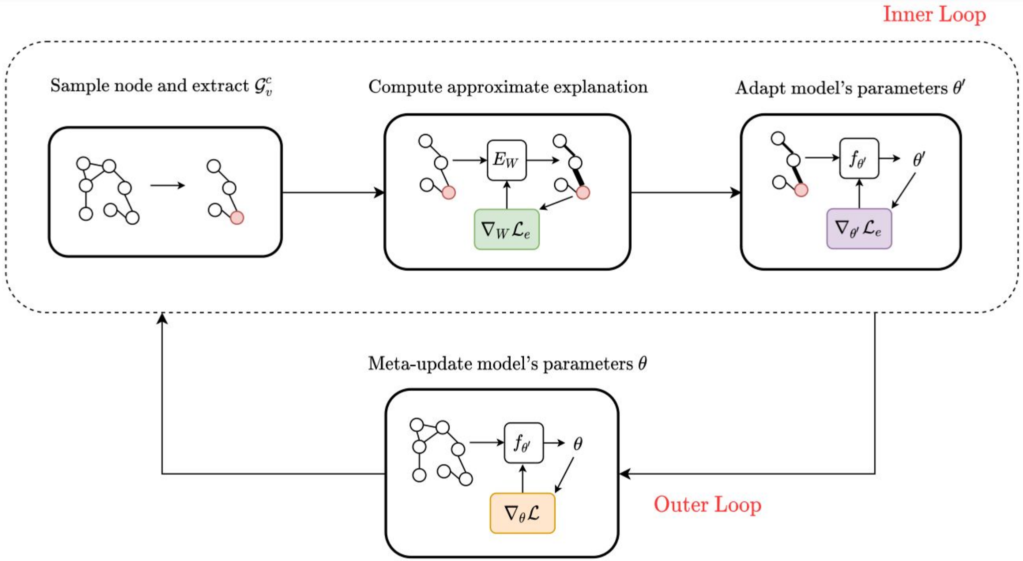
Selected topics

New directions for improving XAI

Challenges for current XAI on graphs

1. **Most interpretability methods are trained post-hoc.**
2. Lack of datasets with an interpretability gold standard.
3. Defining proper metrics for assessing the results is non trivial.

Meta-learning for enhancing XAI



Inner Loop
Sample node and extract $\mathcal{G}_v^c$
Compute approximate explanation
Adapt model's parameters θ'
E_W
$\nabla_W \mathcal{L}_e$
$f_{\theta'}$
θ'
$\nabla_{\theta'} \mathcal{L}_e$
Meta-update model's parameters θ
$f_{\theta'}$
θ
$\nabla_\theta \mathcal{L}$
Outer Loop

Fig. 3. Schematics of our meta-learning framework for improving GNN's explainability at training time. MATE steers the optimization procedure toward more interpretable minima in the inner loop, meanwhile optimizing for the original task in the outer one. The inner loop adapts the model's parameters to a single "explanation task." It starts with the sampling of a random node and its computational subgraph. Then, we train GNNExplainer to explain the current model's prediction. Afterward, we can adapt the model's parameters to the "explanation task" ending in a new model's state. Finally, we meta-update the original parameters minimizing the cross-entropy loss computed with the adapted parameters.

Spinelli, I., Scardapane, S. and Uncini, A., 2022. A Meta-Learning Approach for Training Explainable Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems.*

Some results

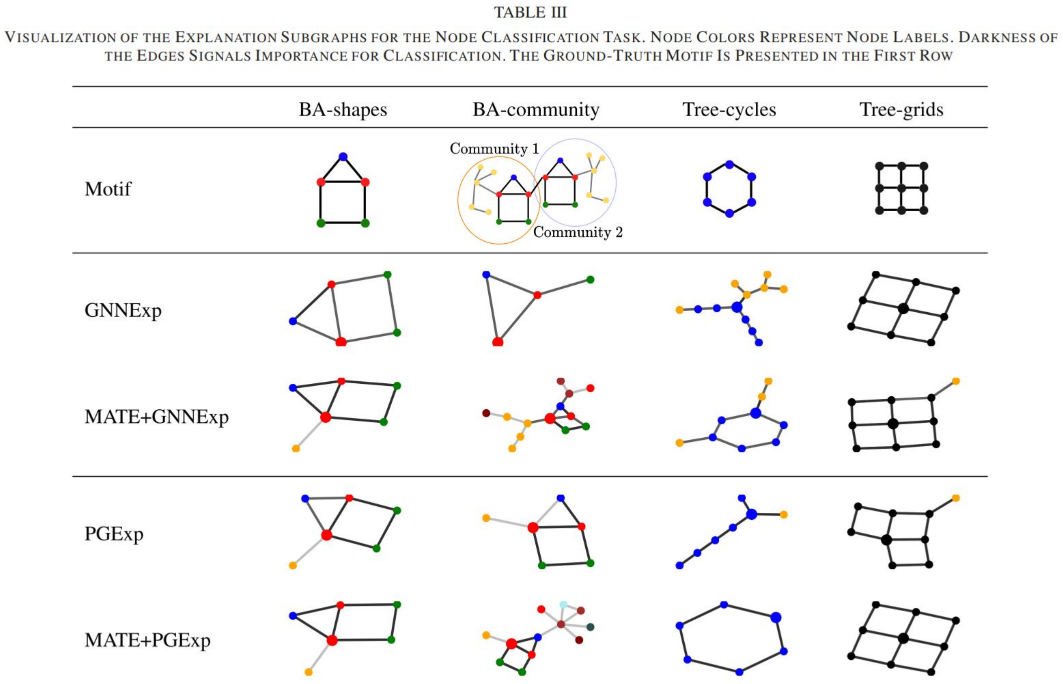


TABLE III

VISUALIZATION OF THE EXPLANATION SUBGRAPHS FOR THE NODE CLASSIFICATION TASK. NODE COLORS REPRESENT NODE LABELS. DARKNESS OF THE EDGES SIGNALS IMPORTANCE FOR CLASSIFICATION. THE GROUND-TRUTH MOTIF IS PRESENTED IN THE FIRST ROW

	BA-shapes	BA-community	Tree-cycles	Tree-grids
Motif		Community 1 Community 2		
GNNExp				
MATE+GNNExp				
PGExp				
MATE+PGExp				

Spinelli, I., Scardapane, S. and Uncini, A., 2022. A Meta-Learning Approach for Training Explainable Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems.*

Prototype-based GNNs

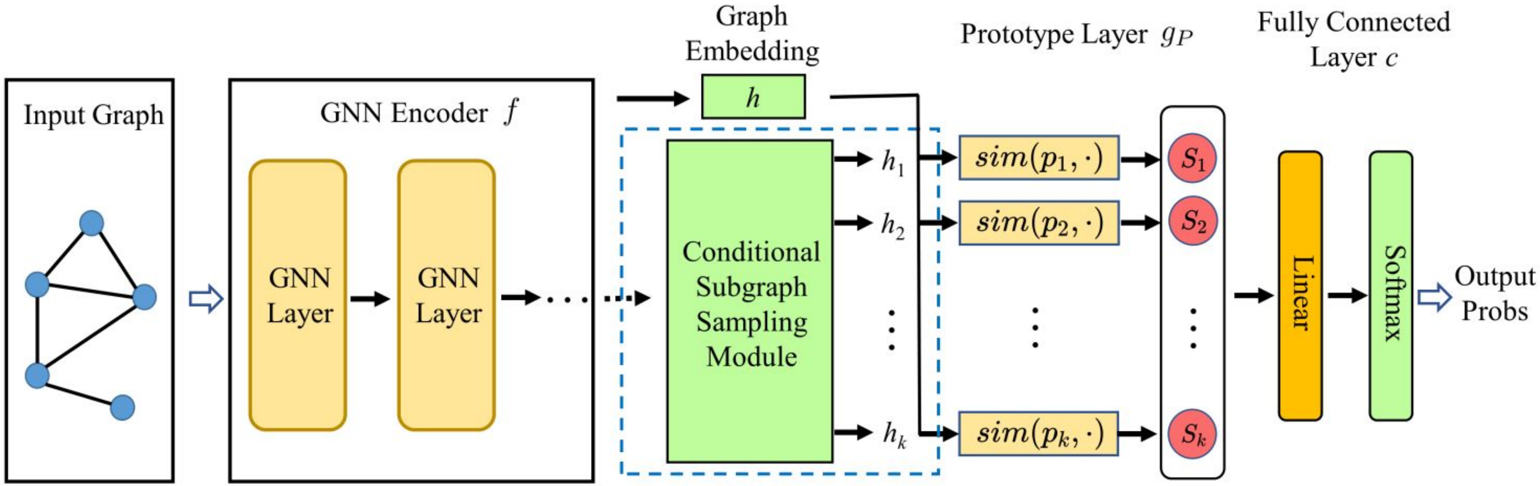


Figure 1: The architecture of our proposed ProtGNN/ProtGNN+. The model mainly consists of three parts: GNN encoder f, prototype layer g_P, and the fully connected layer c appended by softmax to output probabilities in multi-class classification tasks. ProtGNN calculates the similarity score ($sim(p_k, \cdot)$ in the illustration) between the graph embedding and the learned prototypes in the prototype layer. For further interpretability, the conditional subgraph sampling module (in the dashed bounding box) is incorporated in ProtGNN+ to output subgraphs most similar to each learned prototype.

Zhang, Z., Liu, Q., Wang, H., Lu, C., & Lee, C. (2022). **ProtGNN: Towards self-explaining graph neural networks**. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 36, No. 8, pp. 9127-9135).

Thanks! Questions?

Simone Scardapane
Tenure-track Assistant Professor

🌐 https://www.sscardapane.it/

https://twitter.com/s_scardapane