Deep Graph Networks

INTELLIGENT SYSTEMS FOR PATTERN RECOGNITION (ISPR)

DAVIDE BACCIU – DIPARTIMENTO DI INFORMATICA - UNIVERSITA' DI PISA

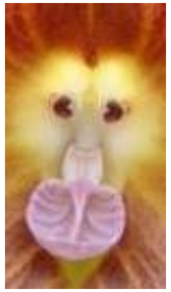
DAVIDE.BACCIU@UNIPI.IT

Lecture Outline

- Motivations
- Formalization of the learning task: graph prediction, induction, transduction and generation
- Historical perspective: contractive and contextual models
- A view on modern deep learning for graphs
- Applications & wrap-up

Introduction

Why Graphs?



Why Graphs?

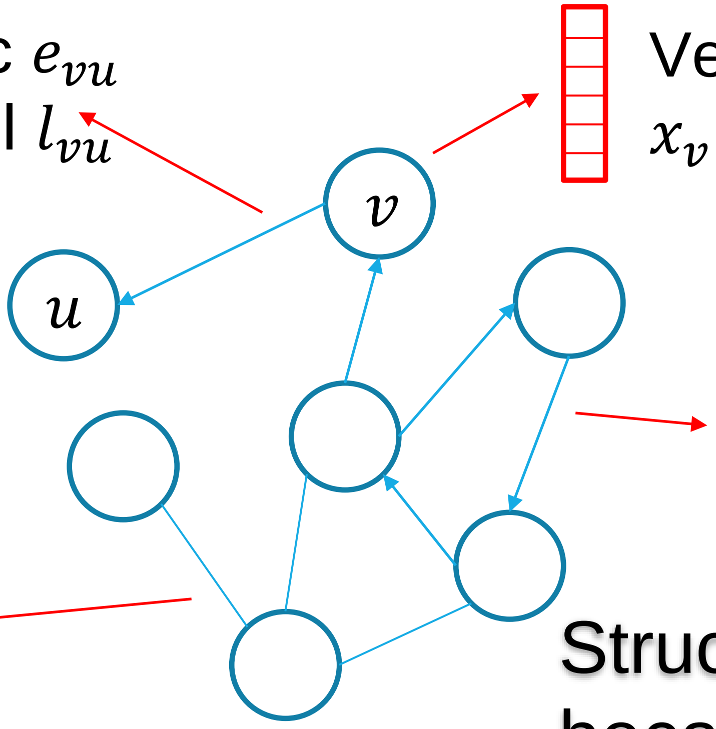
Context is fundamental for the correct interpretation of information

Graph Structured Data

Oriented edge/arc e_{vu}
possibly with label l_{vu}

Vectorial node label
x_v

Node/vertex u



Cycle

Undirected edge

Structures are useful because allow to **represent relationships** in the data

A Nomenclature Nightmare

Neural networks for graphs

Deep learning for graphs

CNN for/on graphs

Graph neural networks

Deep Graph Networks

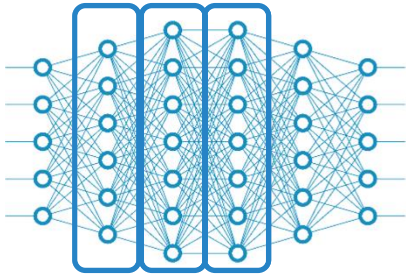
Graph CNN

Learning graph/node embedding

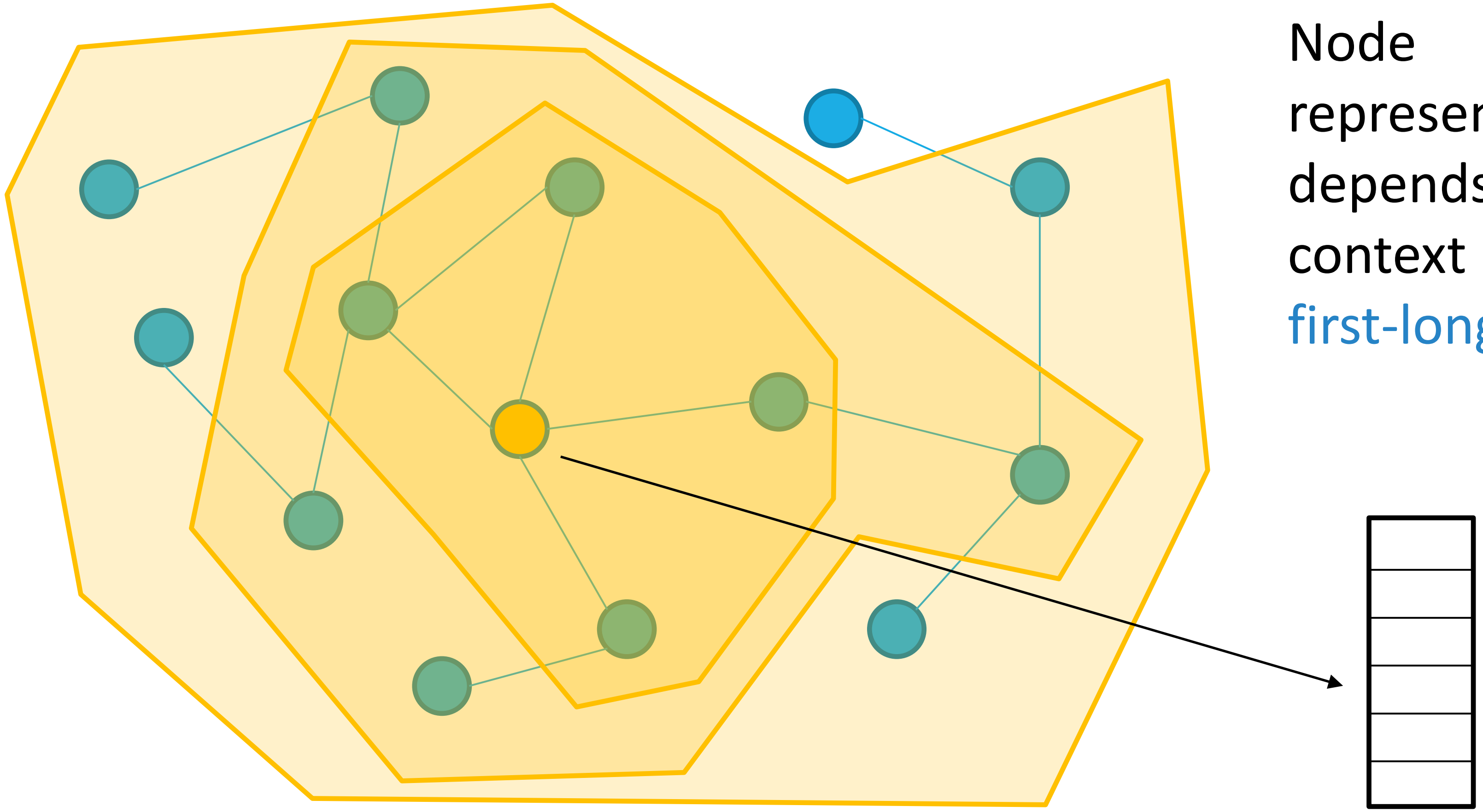
Geometric deep learning

Graph Convolutional Networks

Deep Learning with graphs



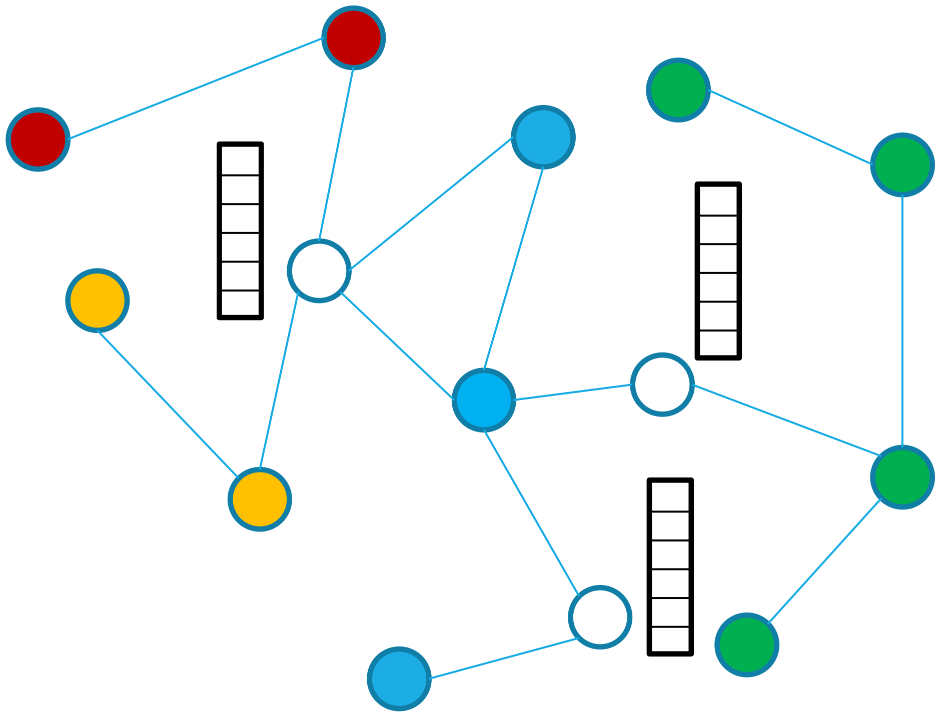
Hierarchical representation learning allows to efficiently diffuse information through graph structure



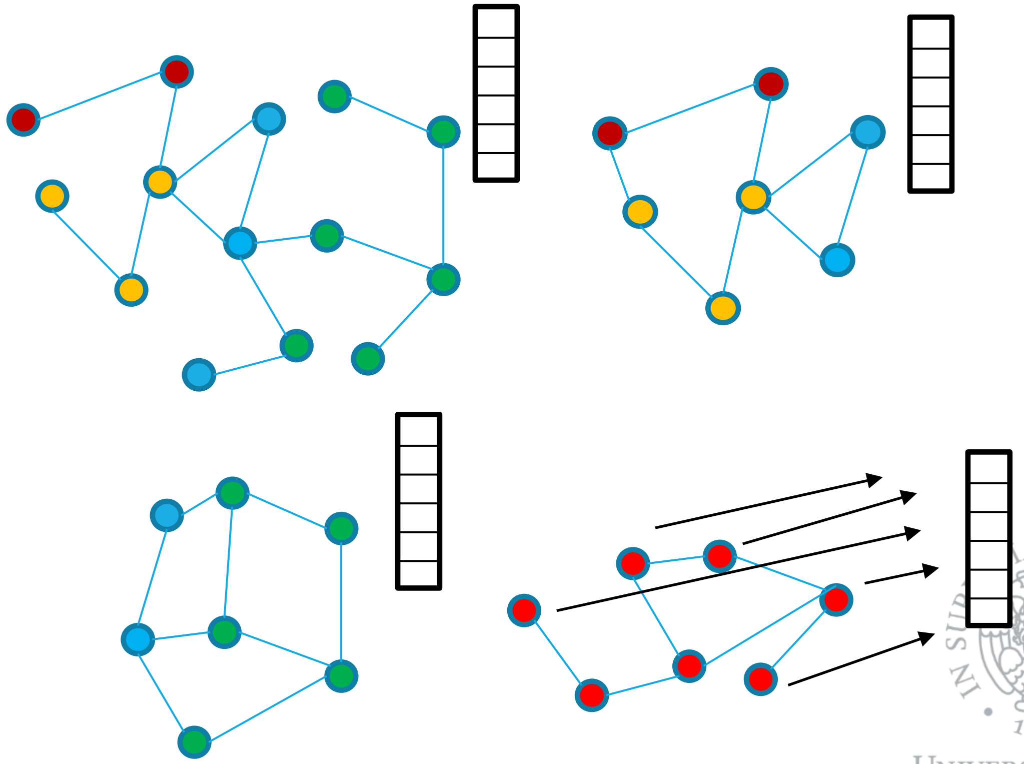
Node representation depends on its context (shorter first-longer later)

Predictive Tasks

Network data

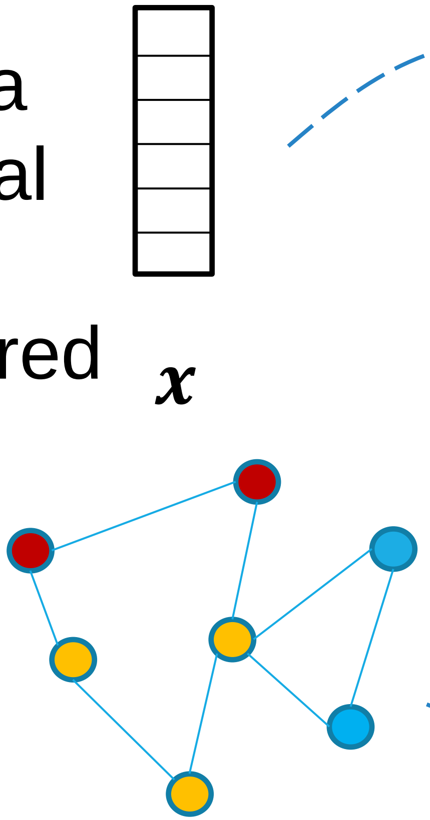


Structure classification/regression



Transductive tasks

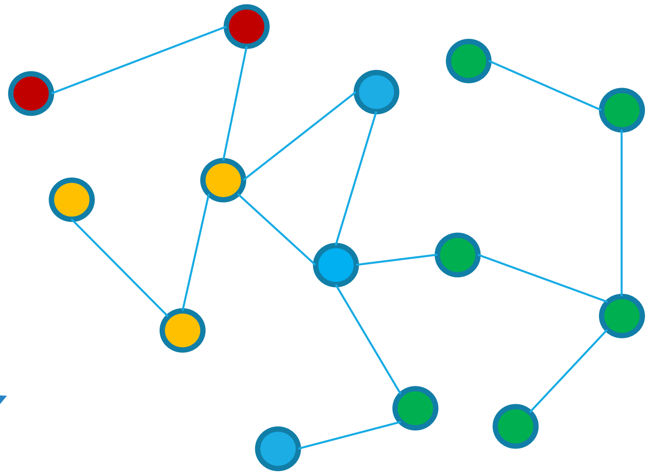
Given a vectorial and/or structured input



$\boldsymbol{x}$

$$\boldsymbol{y} \sim P(\boldsymbol{y}|\boldsymbol{x})$$

$\boldsymbol{y}$



Learn to generate a structured prediction

UNIVERSITÀ DI PISA

An Hystorical (and Geographical) Perspective

Early neural network approaches to deal with cyclic graphs of varying topology date back to 2005-2009

(Sperduti & Starita, TNN 1997)

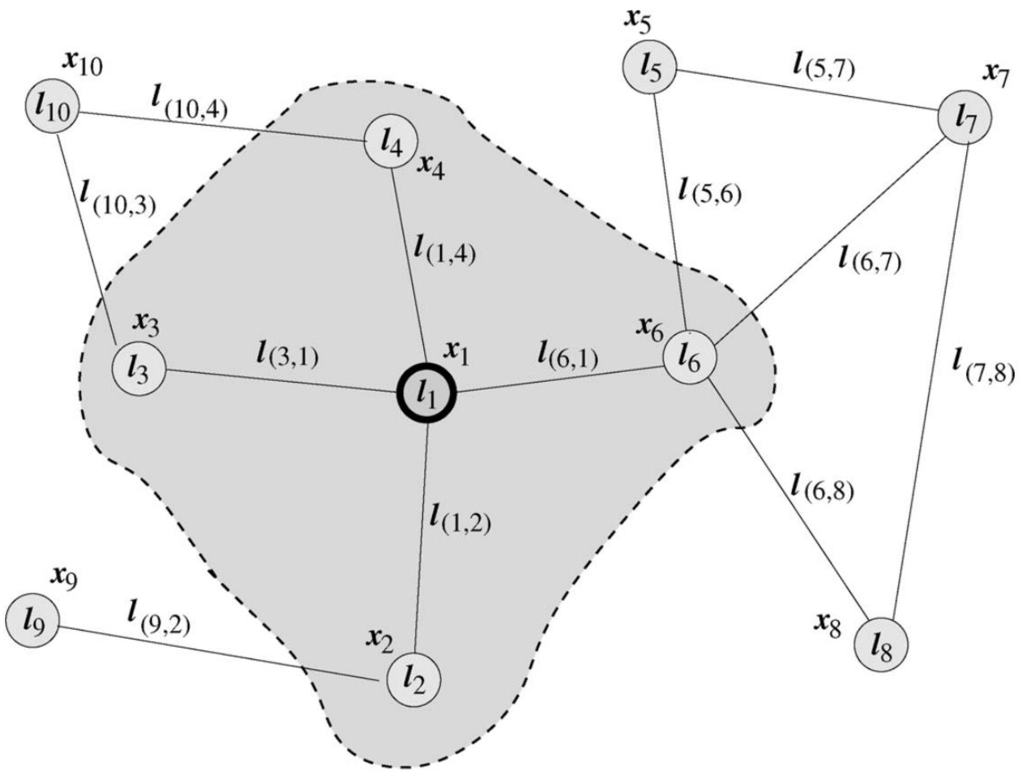
A. Micheli, TNN 2009

1343
UNIVERSITÀ DI PISA

Scarselli et al, TNN 2009

UNIVERSITÀ DI SIENA
1240

Contractive - Graph Neural Networks (GNN)



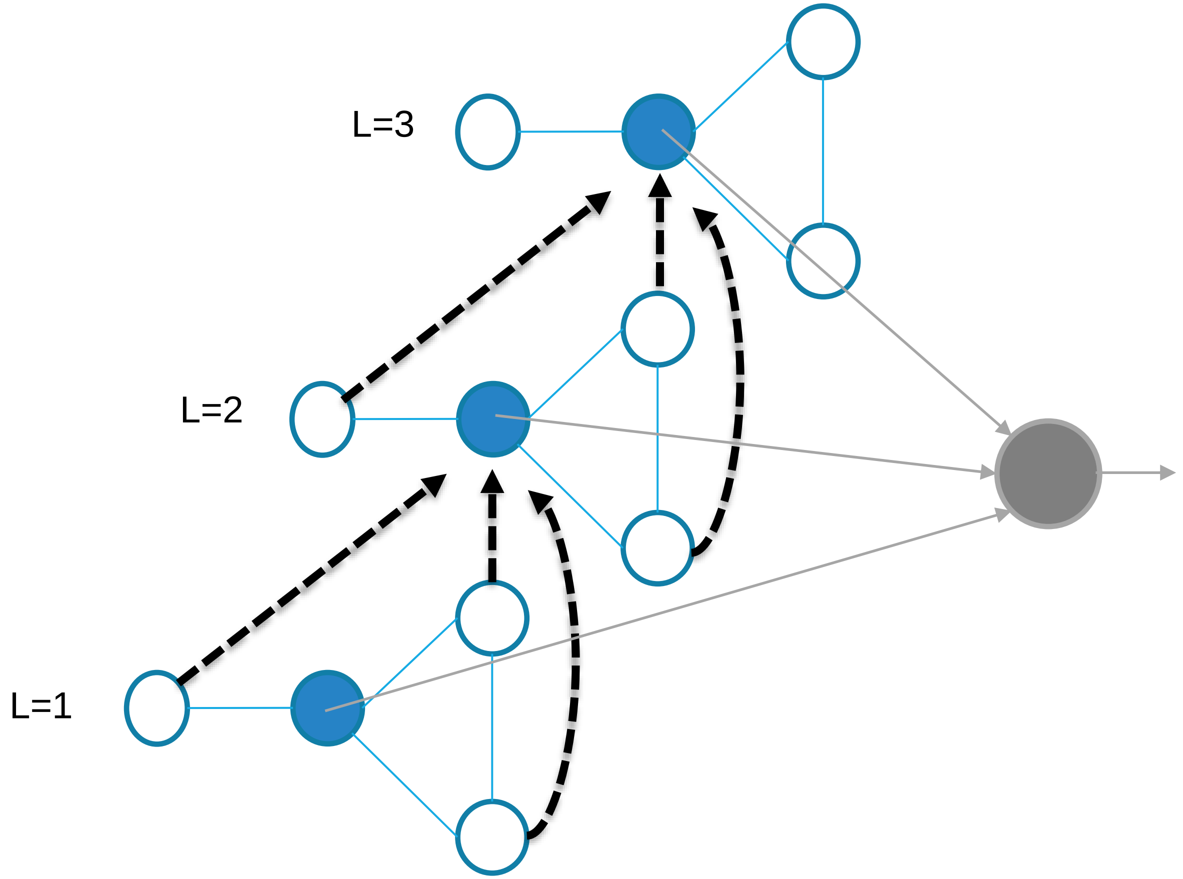
$$x_1 = f_w(l_1, \underbrace{l_{(1,2)}, l_{(3,1)}, l_{(1,4)}, l_{(6,1)}}_{l_{co[1]}}, \underbrace{x_2, x_3, x_4, x_6}_{x_{ne[1]}}, \underbrace{l_2, l_3, l_4, l_6}_{l_{ne[n]}})$$

- Extend the Recurrent/Recursive Neural Network approach to cyclic graphs
- Handle loops through fixed points
- Impose dynamic weight constraints to yield a contractive state mapping

Scarselli et al, TNN 2009

https://sailab.diism.unisi.it/gnn/

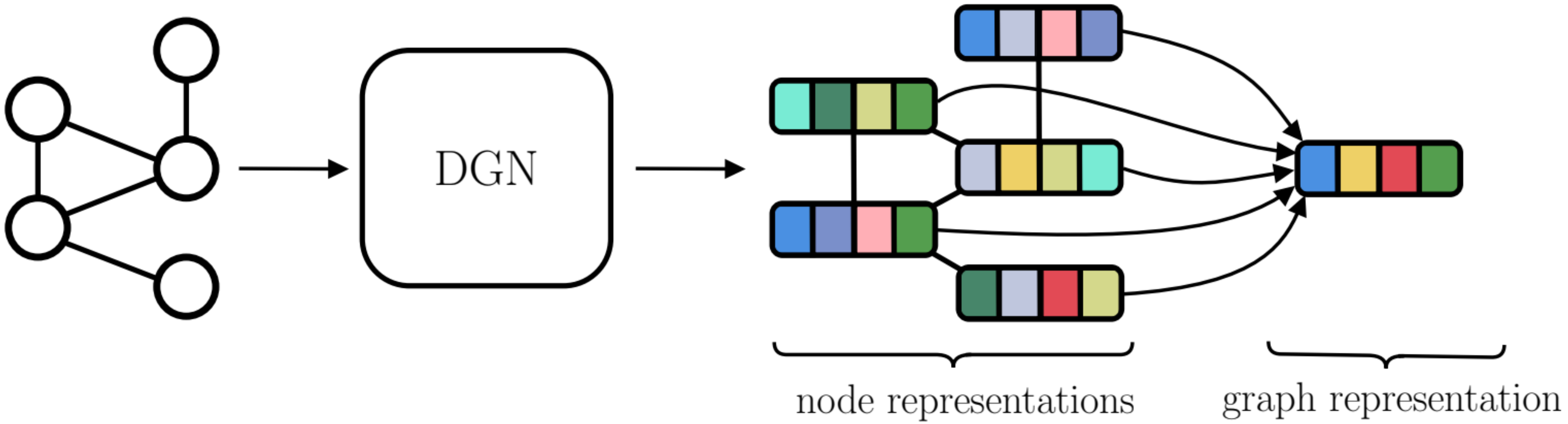
Contextual - Neural Networks for Graphs (NN4G)



- A feedforward approach to process graphs
- Handle loops through layering
- Uses context from frozen earlier layers compute the state on the node at current layer
- Layerwise training

A. Micheli, TNN 2009

Deep Graph Networks



- Encode vertices and the graph itself into a vector space by means of an adaptive (learnable) mapping
- Use the learned encodings to solve predictive and explorative tasks

A Survey of Recent Approaches

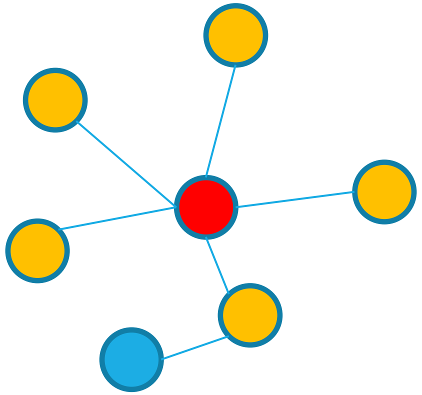
- Convolutional Neural Networks for Graphs
  - Spectral
  - Spatial
- Contextual Graph Processing
  - Contextual Graph Convolutions
  - Node embeddings
- Generative approaches

Convolutional Neural Networks for Graphs

How to Perform Convolutions on Graphs?

SPATIAL DOMAIN

What is the equivalent of sliding a kernel to aggregate local spatial information?



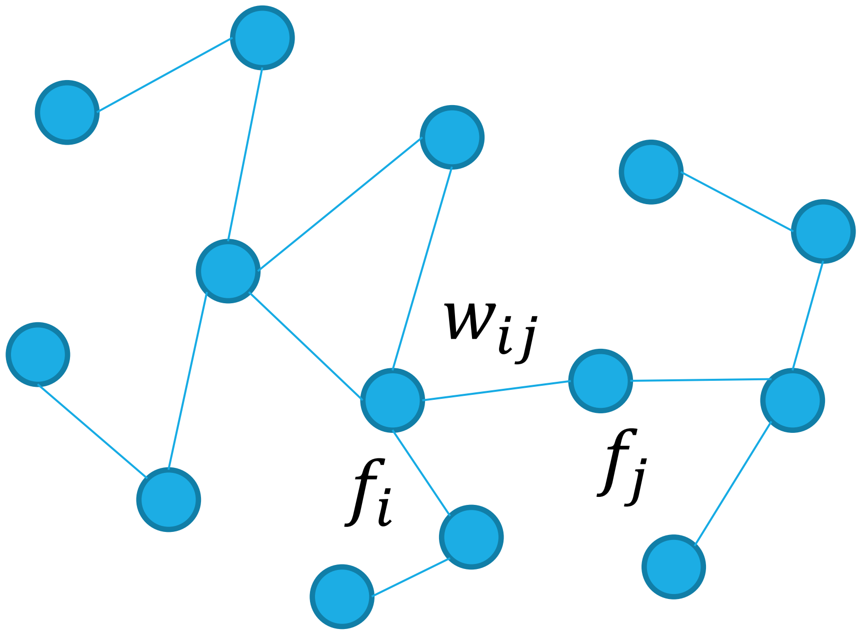
SPECTRAL DOMAIN

$$\mathcal{F}(f * g) = \mathcal{F}(f) \times \mathcal{F}(g)$$

Exploit the Convolution Theorem and Fourier analysis to perform convolutions in the spectral domain

Decompose a function f as a combination of vectors $\boldsymbol{e}_k$ from an orthonormal basis

The Spectral Scenario



- Single weighted undirected graph
  - $w_{ij} > 0$ weight of the i-j edge
- Functions f_i attaching values (i.e. labels/signals x_i) to nodes i
- Task: process the signals defined on the graph structure

IN SUPREMÆ DIGNITATIS 1343
UNIVERSITÀ DI PISA

Spectral Graph Convolution in 1 Slide

- Given a graph G, the eigendecomposition of its Laplacian provides an orthonormal basis U which allow to compute the graph convolution of its node signals $\boldsymbol{f}$ with a filter

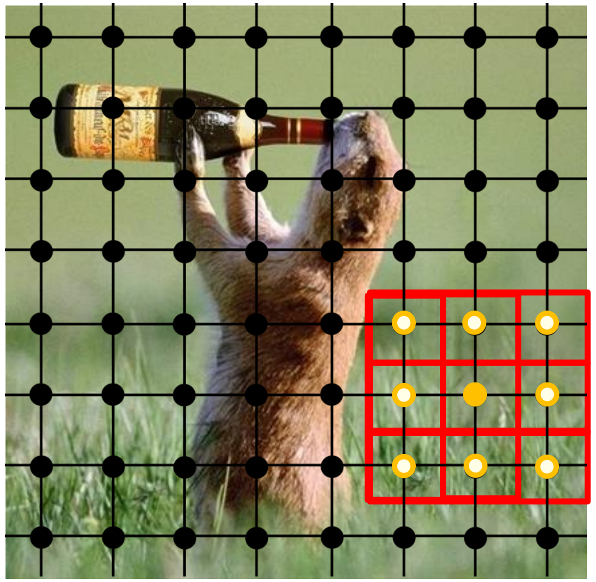
$$(\boldsymbol{f} *_G \boldsymbol{g}) = \mathcal{F}^{-1}\big(\mathcal{F}(\boldsymbol{f})\,\mathcal{F}(\boldsymbol{g})\big) = U\mathbf{W}(\lambda)U^T\boldsymbol{f}$$

Convolutional filter **g** in spectral domain

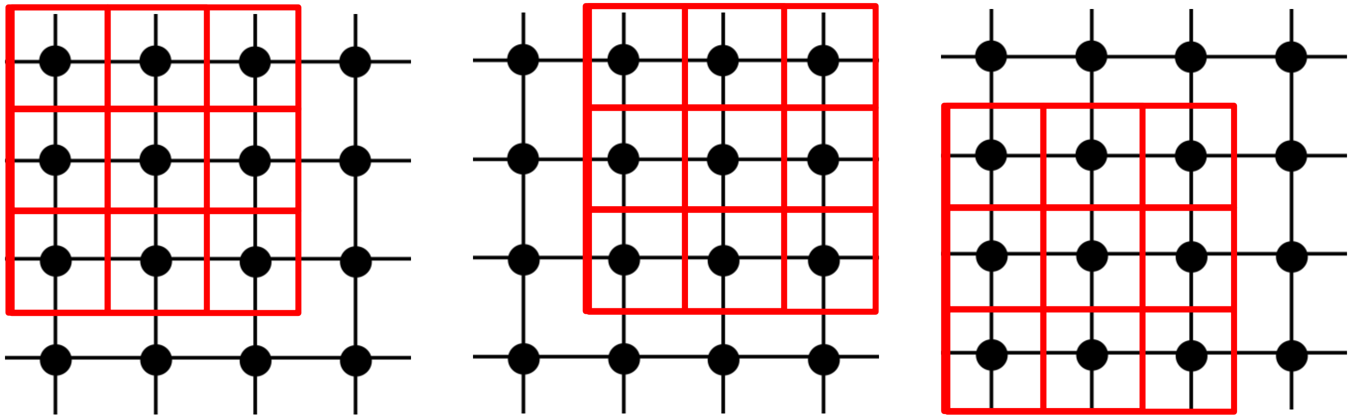
Graph equivalent of the learnable CNN filter matrix **W**

Spectral convolution matrix **W** contains information on the graph Laplacian

A Graph View on (Image) Convolutions



Visual convolutions are graph convolutions on a regular grid

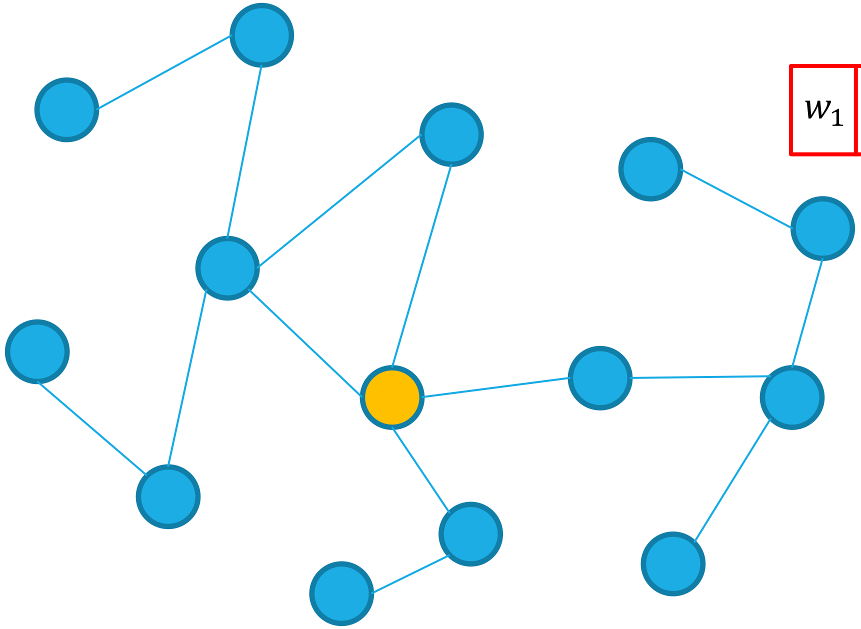


Plus some key assumptions which make it difficult to directly apply them to graphs

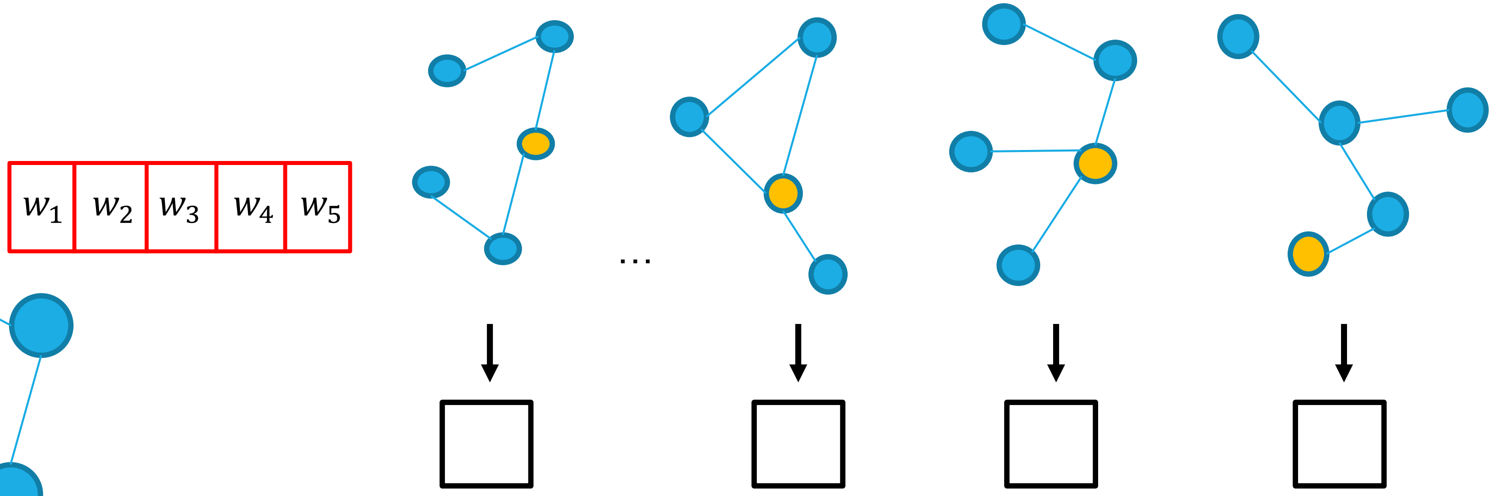
- ❖ Regular neighborhood
- ❖ Existence of a total node ordering

Node Neighborhoods

Example of 4-neighborhoods



convolutions



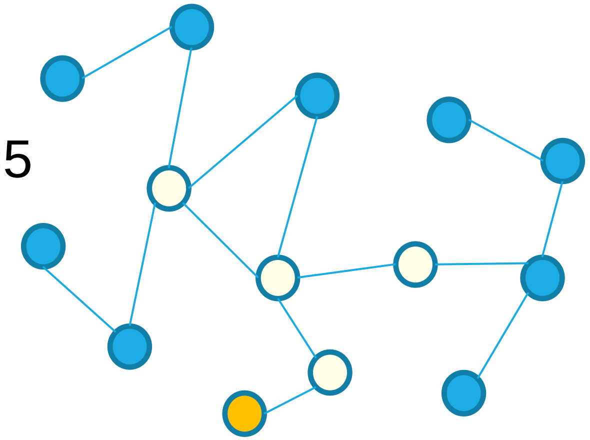
Neighborhoods depend on node ordering: how can I get coherent node ordering across multiple graphs?

PATCHY-SAN

Niepert, Ahmed, Kutzkov, ICML 2016

Leverage graph labelling techniques (e.g. Weisfeiler-Lehman) to determine a coherent ordering within the graph and between the graphs

Neighborhood for k=5



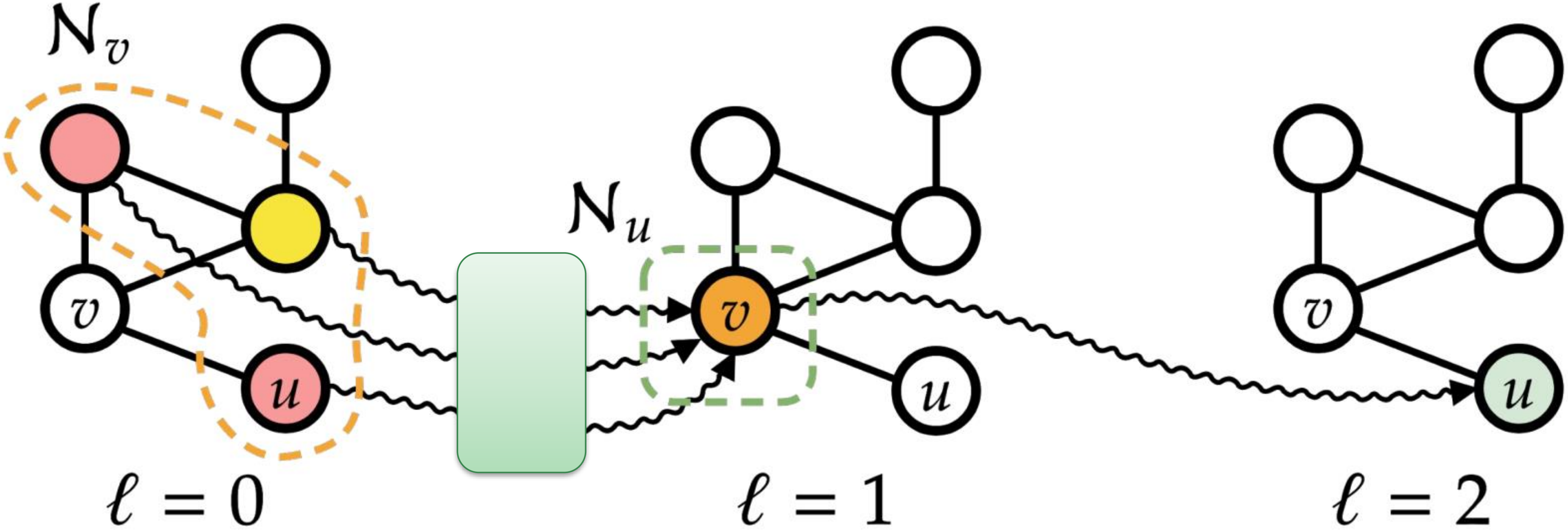
Parametric convolutional filter of size k

w_1	w_2	w_3	w_4	w_5

Determining a coherent ordering to match nodes to filter parameters in NP complete (graph normalization)

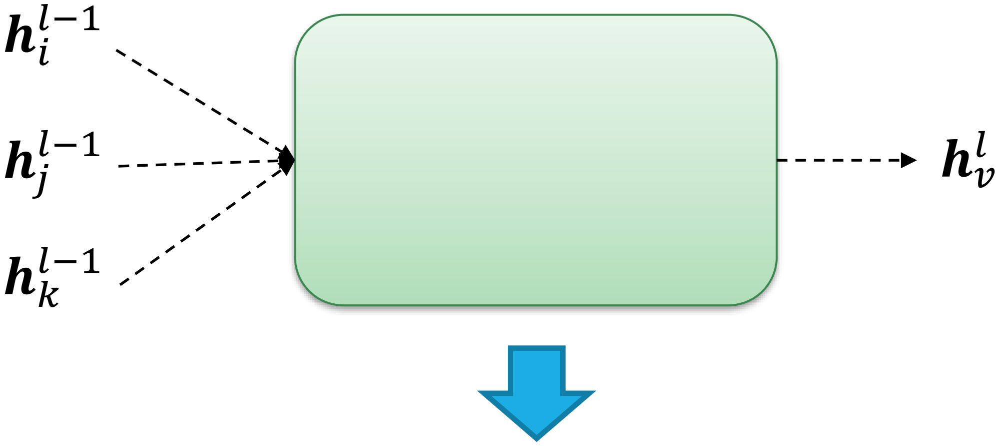
Contextual Graph Processing

Neighborhood Aggregation & Layering



What is inside of the Box?

A learning model of course (e.g. a neural network) including an aggregation function to handle size-varying neighborhoods



A simple model

$$\boldsymbol{h}_v^{\mathrm{l}} = \sigma\left(\boldsymbol{W}_l AGG\left(\left\{\boldsymbol{h}_i^{l-1}: i \in N(v)\right\}\right), \widehat{\boldsymbol{W}}_l \boldsymbol{h}_v^{l-1}\right)$$

The graph convolutional layer

MLP/Linear

$$\underbrace{\mathbf{h}_v^{\ell+1}}_{\text{state}} = \overbrace{\phi^{\ell+1}}\Big(\mathbf{h}_v^{\ell},\ \underbrace{\Psi}_{\text{perm. invariant function}}\big(\{\overbrace{\psi^{\ell+1}}(\mathbf{h}_u^{\ell}) \mid u \in \mathcal{N}_v\}\big)\Big)$$

Variants/extensions:

- **Edge-aware convolution**
- **Attention over neighbors**
- **Laplacian-normalized**

Model	Neighborhood Aggregation $\mathbf{h}_v^{\ell+1}$
NN4G [88]	$\sigma\Big(\mathbf{w}^{\ell+1^T}\mathbf{x}_v + \sum_{i=0}^{\ell}\sum_{c_k\in\mathcal{C}}\sum_{u\in\mathcal{N}_v^{c_k}} w_{c_k}^i * \mathbf{h}_u^i\Big)$
GNN [104]	$\sum_{u\in\mathcal{N}_v} MLP^{\ell+1}\Big(\mathbf{x}_u, \mathbf{x}_v, \mathbf{a}_{uv}, \mathbf{h}_u^{\ell}\Big)$
GraphESN [44]	$\sigma\Big(\mathbf{W}^{\ell+1}\mathbf{x}_u + \hat{\mathbf{W}}^{\ell+1}[\mathbf{h}_{u_1}^{\ell}, \ldots, \mathbf{h}_{u_{\mathcal{N}_v}}^{\ell}]\Big)$
GCN [72]	$\sigma\Big(\mathbf{W}^{\ell+1}\sum_{u\in\mathcal{N}(v)}\mathbf{L}_{vu}\mathbf{h}_u^{\ell}\Big)$
GAT [120]	$\sigma\Big(\sum_{u\in\mathcal{N}_v}\alpha_{uv}^{\ell+1} * \mathbf{W}^{\ell+1}\mathbf{h}_u\Big)$
ECC [111]	$\sigma\Big(\frac{1}{\lvert\mathcal{N}_v\rvert}\sum_{u\in\mathcal{N}_v} MLP^{\ell+1}(\mathbf{a}_{uv})^T\mathbf{h}_u^{\ell}\Big)$
R-GCN [105]	$\sigma\Big(\sum_{c_k\in\mathcal{C}}\sum_{u\in\mathcal{N}_v^{c_k}}\frac{1}{\lvert\mathcal{N}_v^{c_k}\rvert}\mathbf{W}_{c_k}^{\ell+1}\mathbf{h}_u^{\ell} + \mathbf{W}^{\ell+1}\mathbf{h}_v^{\ell}\Big)$
GraphSAGE [54]	$\sigma\Big(\mathbf{W}^{\ell+1}(\frac{1}{\lvert\mathcal{N}_v\rvert}[\mathbf{h}_v^{\ell}, \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^{\ell}])\Big)$
CGMM [3]	$\sum_{i=0}^{\ell} w^i * \Big(\sum_{c_k\in\mathcal{C}} w_{c_k}^i * (\frac{1}{\lvert\mathcal{N}_v^{c_k}\rvert}\sum_{u\in\mathcal{N}_v^{c_k}}\mathbf{h}_u^i)\Big)$
GIN [131]	$MLP^{\ell+1}\Big((1+\epsilon^{\ell+1})\mathbf{h}_v^{\ell} + \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^{\ell}\Big)$

A Message-Passing view on Deep Graph Networks

Algorithm 13.1: Simple message-passing neural network

Input: Undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

Initial node embeddings $\{\mathbf{h}_n^{(0)} = \mathbf{x}_n\}$

$\mathrm{Aggregate}(\cdot)$ function

$\mathrm{Update}(\cdot, \cdot)$ function

Output: Final node embeddings $\{\mathbf{h}_n^{(L)}\}$

```
// Iterative message-passing
```

for $l \in \{0, \ldots, L-1\}$ **do**

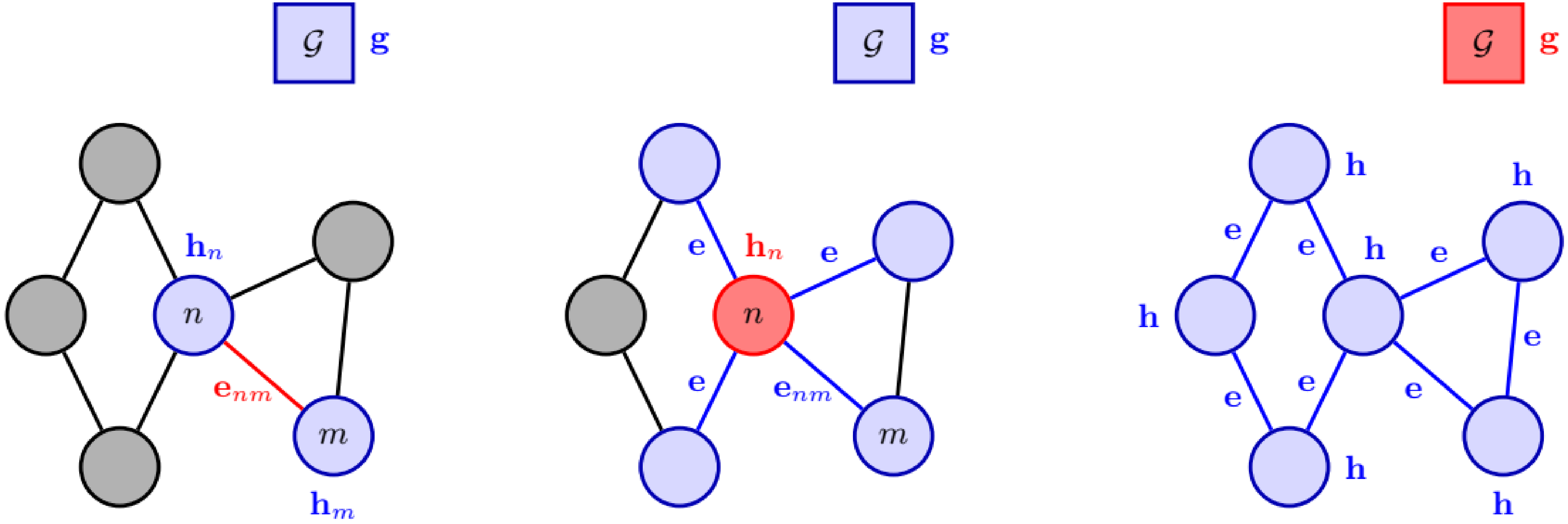
$\quad \mathbf{z}_n^{(l)} \leftarrow \mathrm{Aggregate}\left(\left\{\mathbf{h}_m^{(l)} : m \in \mathcal{N}(n)\right\}\right)$

$\quad \mathbf{h}_n^{(l+1)} \leftarrow \mathrm{Update}\left(\mathbf{h}_n^{(l)}, \mathbf{z}_n^{(l)}\right)$

end for

return $\{\mathbf{h}_n^{(L)}\}$

Different kinds of message-passing updates



Edge **Node** **Graph**

Graph Isomorphism Network (a.k.a. sum is better)

Xu et al, ICLR 2019

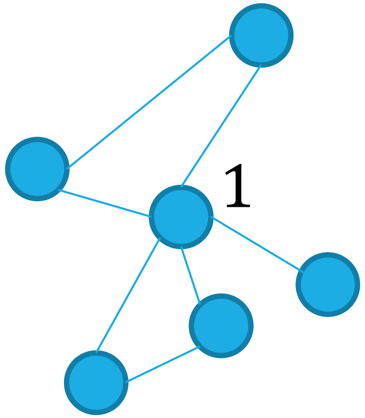
- A study of GNN expressivity w.r.t. WL test of graph isomorphism
- Choice of aggregation functions influences what structures can be recognized
- Propose a simple aggregation and concatenation model

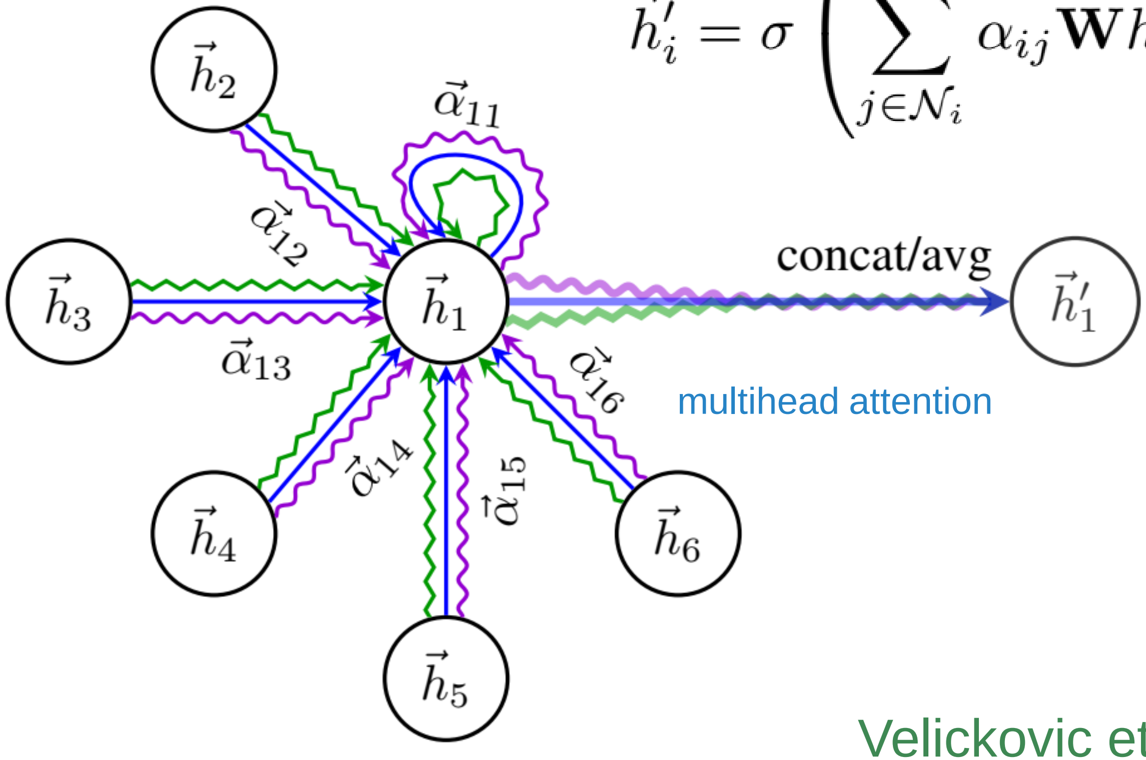
$$h_v^{(k)} = \mathrm{MLP}^{(k)}\left(\left(1 + \epsilon^{(k)}\right) \cdot h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)} \right)$$

$$h_G = \mathrm{CONCAT}\left(\mathrm{READOUT}\left(\{h_v^{(k)} | v \in G\}\right) | k = 0, 1, \cdots, K\right)$$

UNIVERSITÀ DI PISA

Graph Attention





$$\vec{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}\vec{h}_j\right)$$

multihead attention

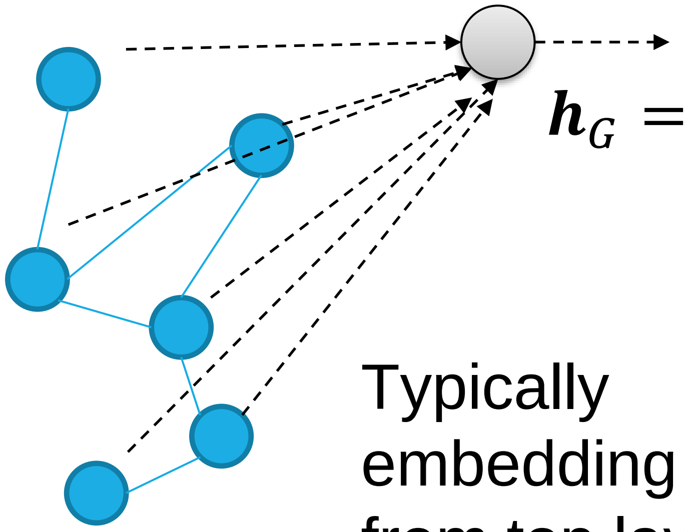
Learning to weight contribution of other nodes when aggregating to form the node embedding

Velickovic et al, ICLR 2018

UNIVERSITÀ DI PISA

Using Node Embedding

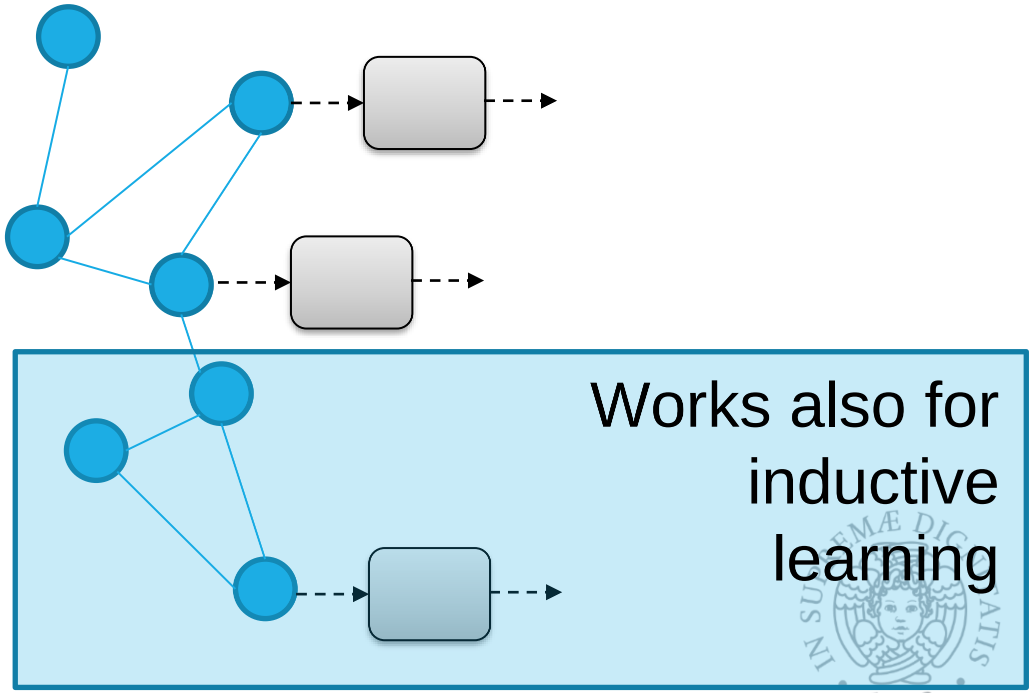
Aggregate all node embeddings to compute graph level predictions



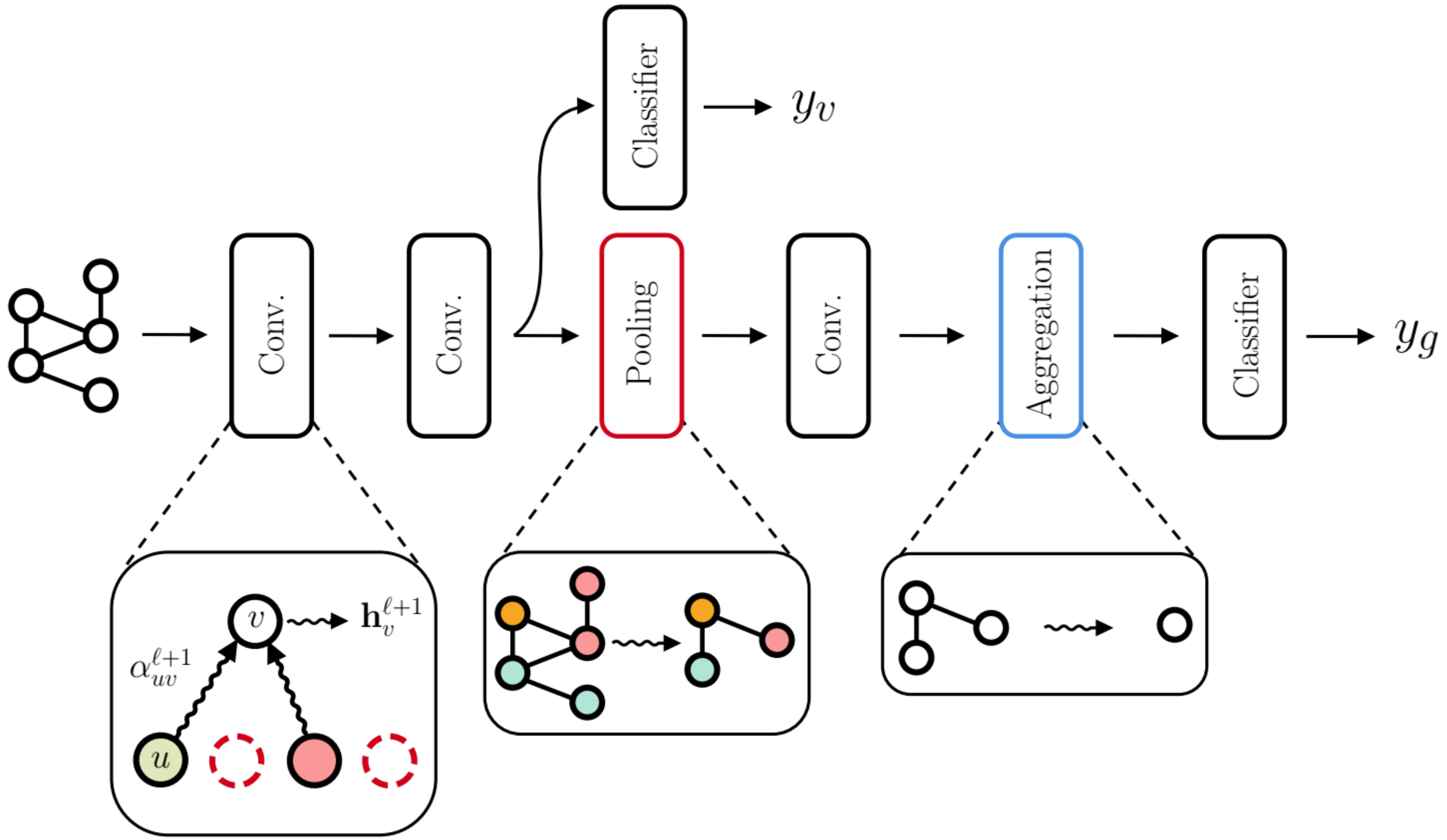
$$\boldsymbol{h}_G = \sum_{i \in G} \boldsymbol{h}_i^L$$

Typically embedding from top layer

Train node level predictors



Deep Graph Networks - The Complete Picture

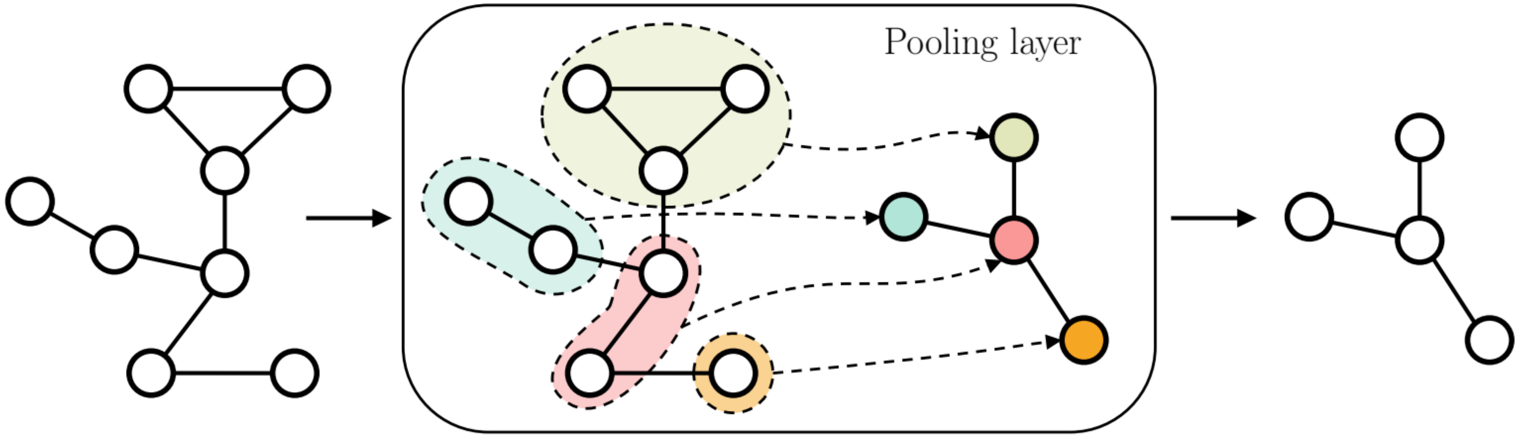


What About Pooling?

- Standard aggregation operates of predefined node subsets
- Ignore community/hierarchical structure in the graph
- Need graph coarsening (pooling) operators
  - Differentiable
  - Graph theoretical
  - Graph signature

Rex Ying et al, NIPS 2018

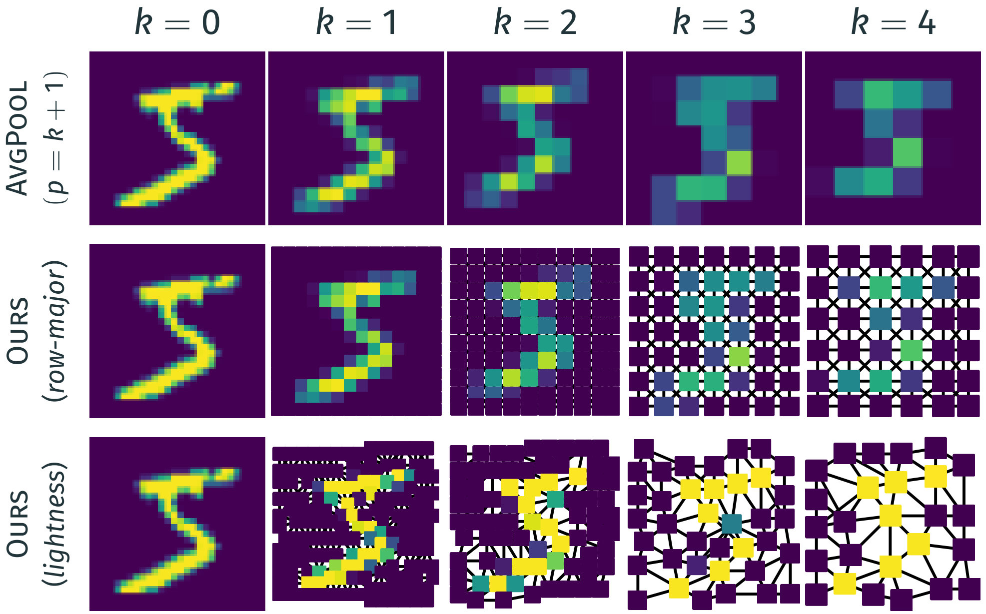
Bacciu et al, AAAI 2023

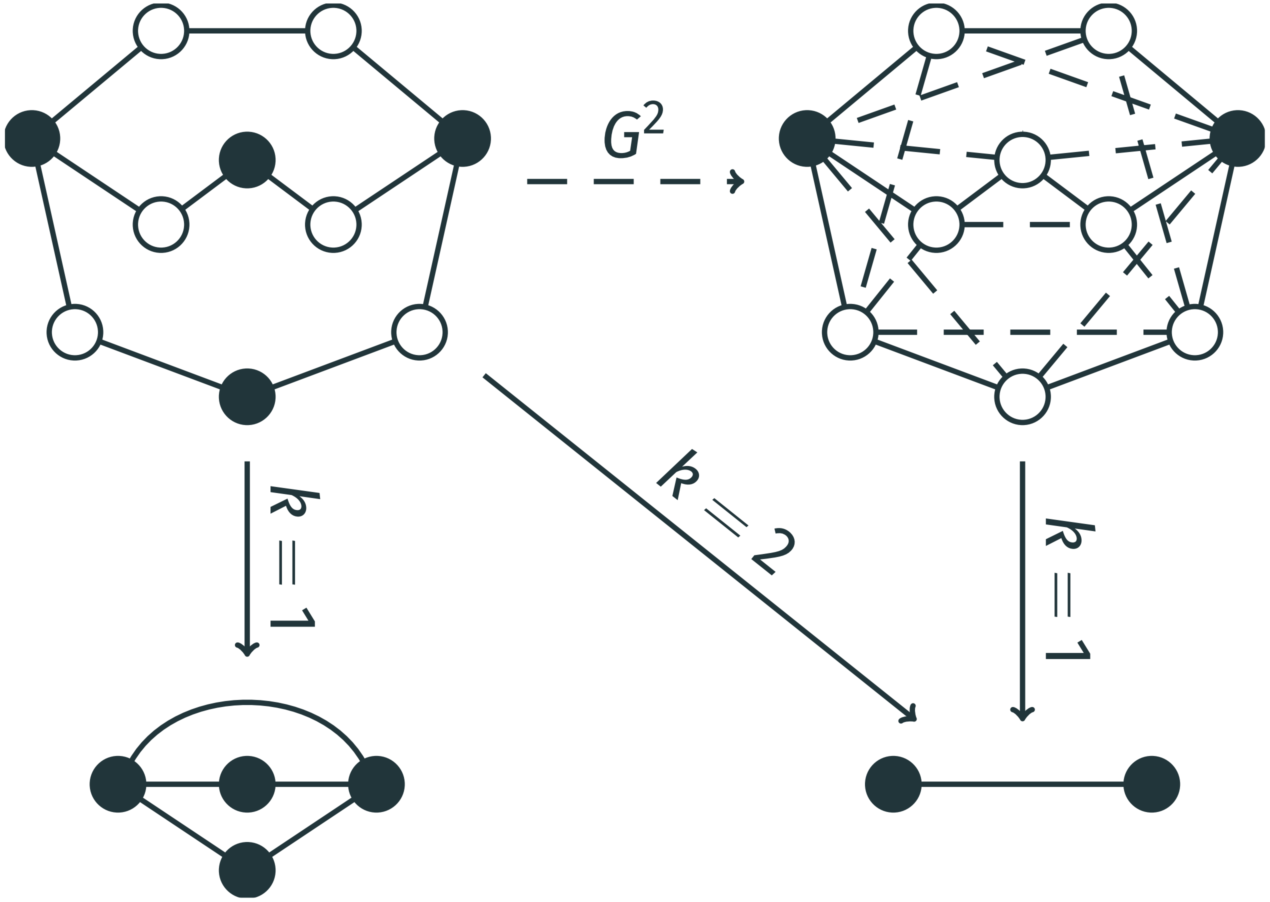


K-MIS Graph Coarsening

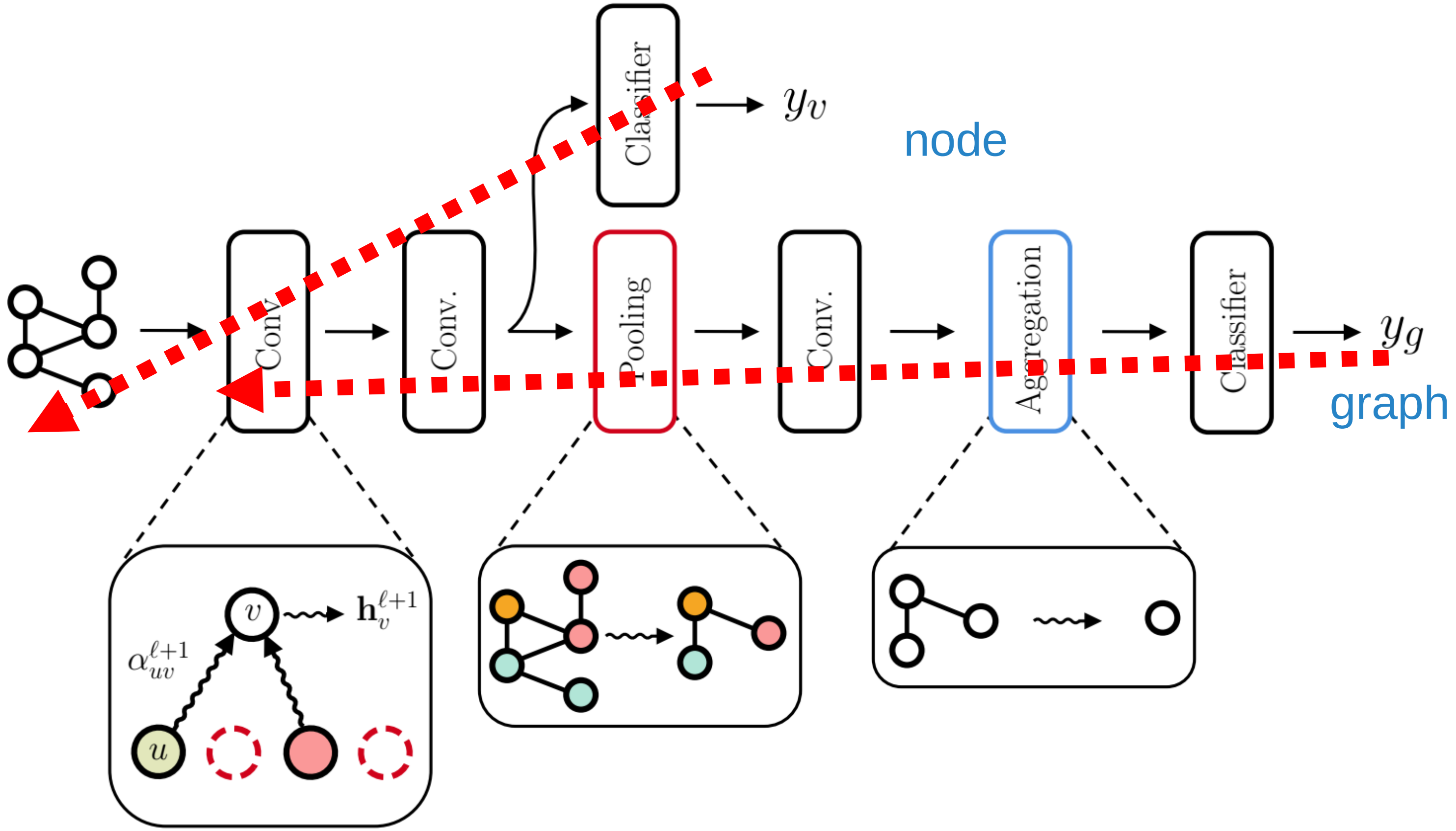
Bacciu et al, AAAI 2023

A proper extension of image-pooling to graphs with theoretical guarantees and scalability





Training the Embedding

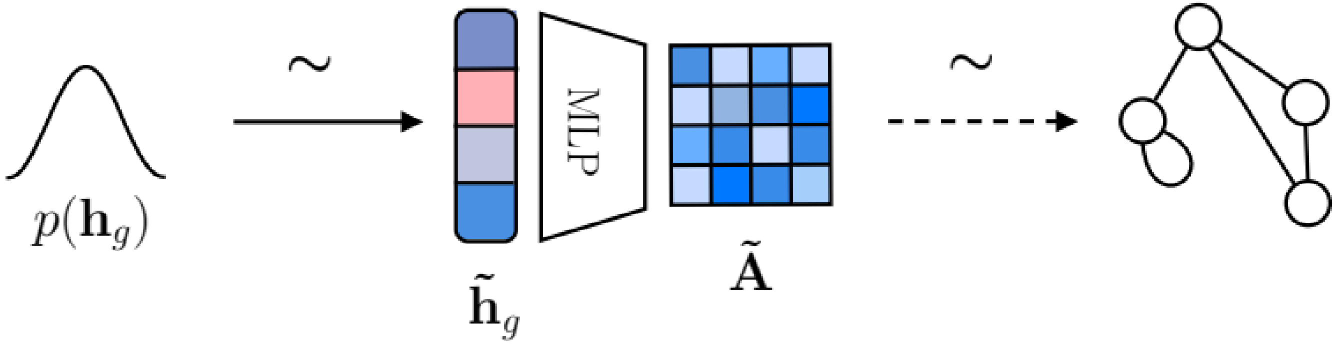


Backpropagate from the (graph or node level) error computed from the top layer embeddings to the early layers

Generative Approaches

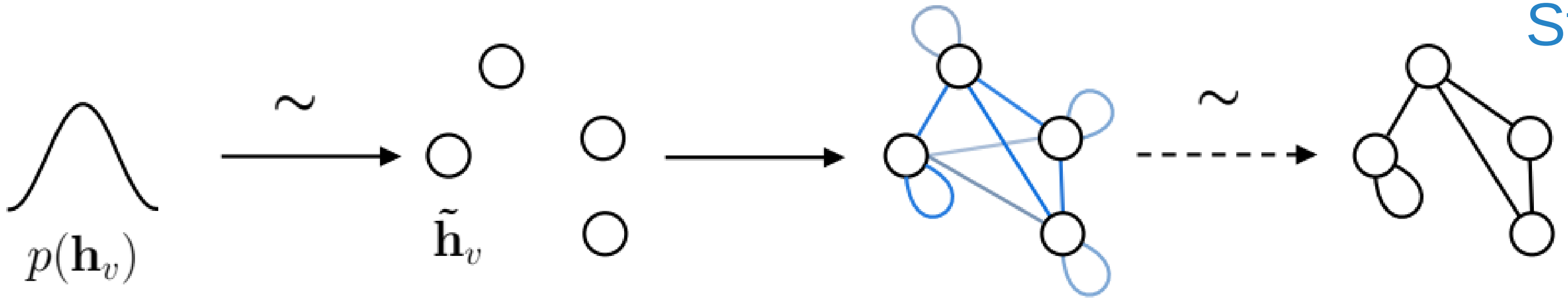
Graph Generation

Generate a prediction that is itself a graph

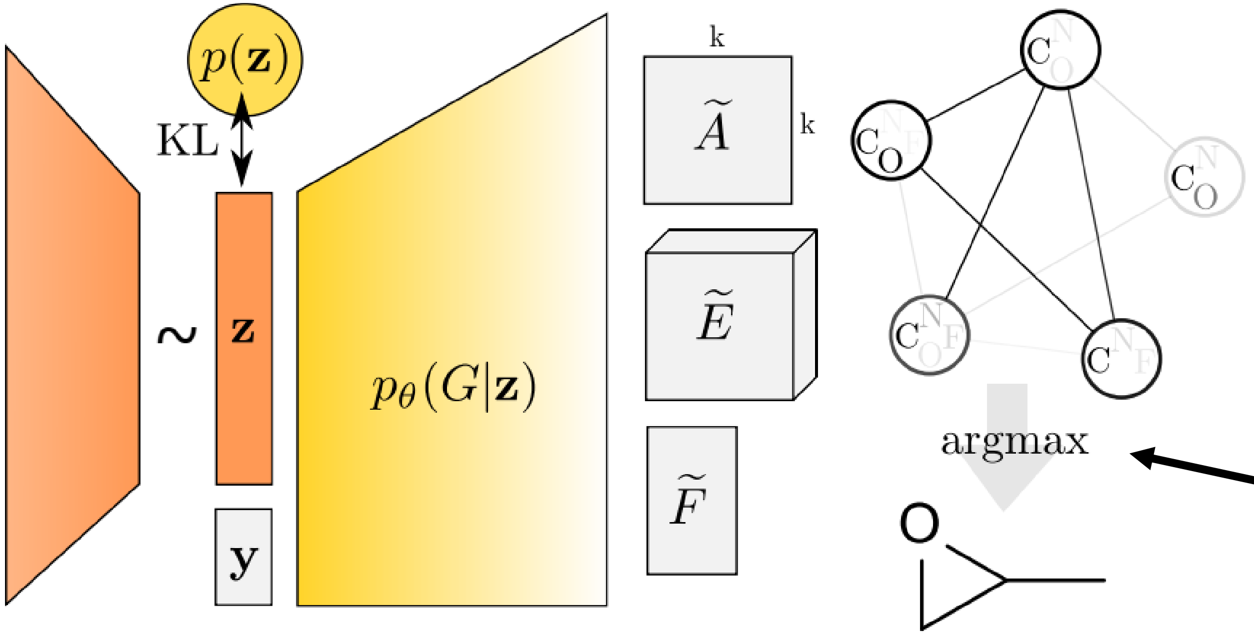


Adjacency-based

Structure-based



Graph Variational Autoencoder

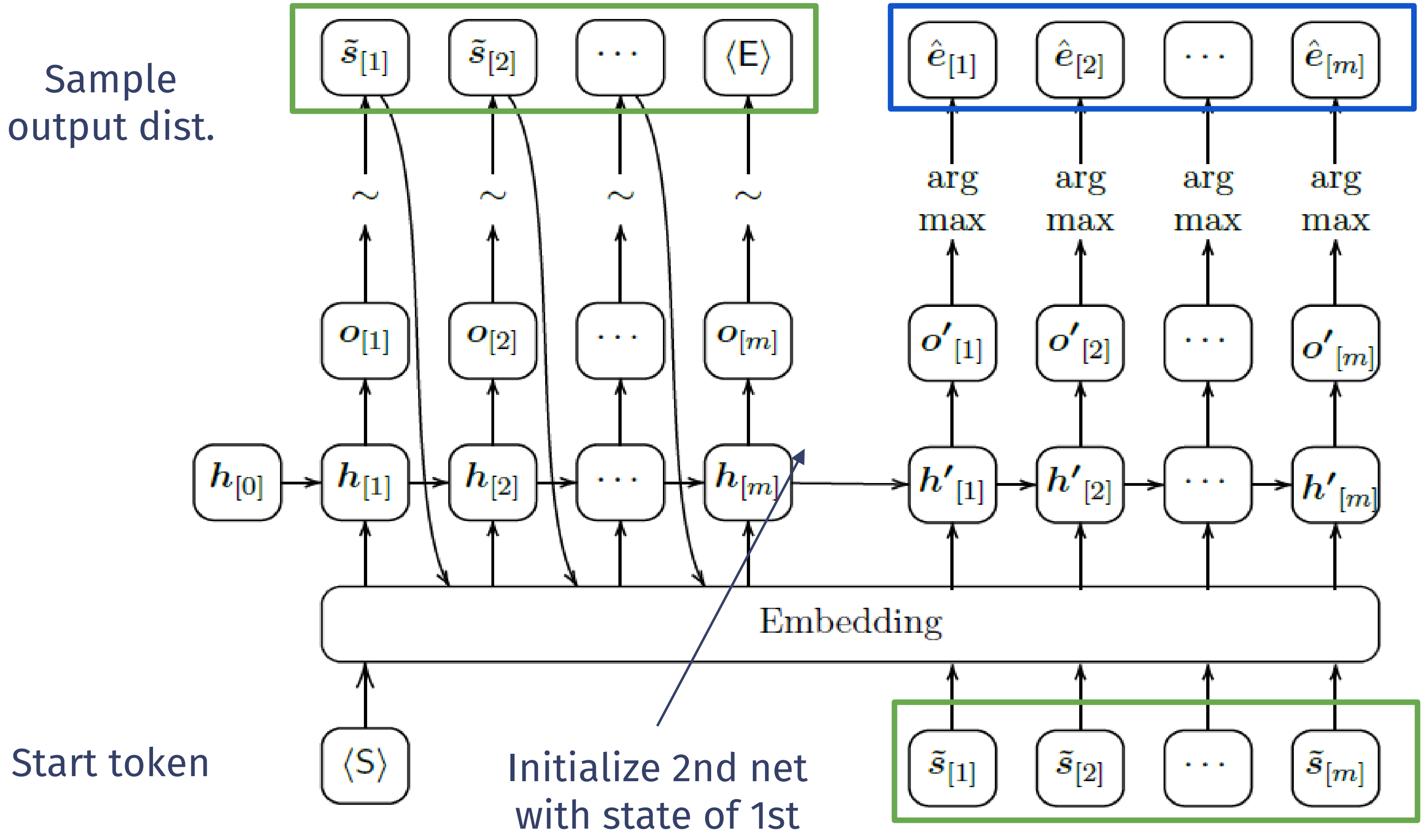


GraphVAE generates adjacency matrix up to k vertices along with the relevant edge/node features (for molecular data)

Argmax a.k.a. sampling $\Longrightarrow$ non-differentiable

Simonovsky, Komodakis, ICLR-WS 2018

Language-Based Graph Generation

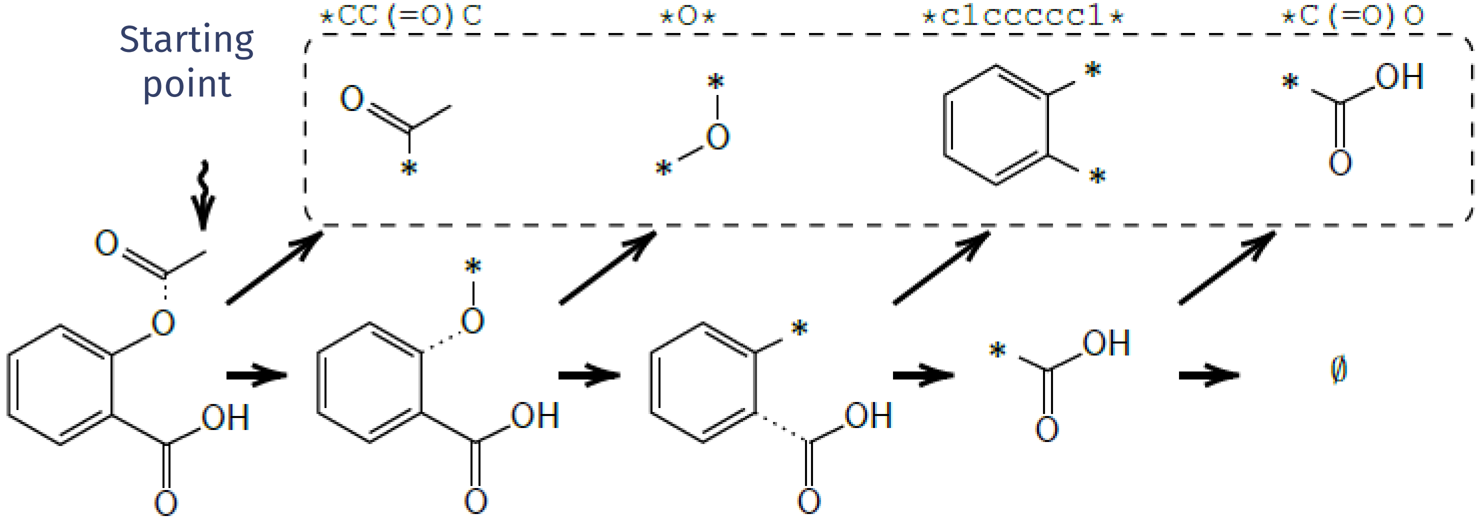


Generate a graph node-by-node and edge-by-edge through a sequential approach

Bacciu, Micheli, Podda, Neurocomputing 2020

Generate Molecules by Fragmentation

- ✓ Molecule is scanned in SMILES order
- ✓ Find first breakable bond
- ✓ Break the molecule at that bond, set aside leftmost fragment
- ✓ Proceed recursively on rightmost fragment

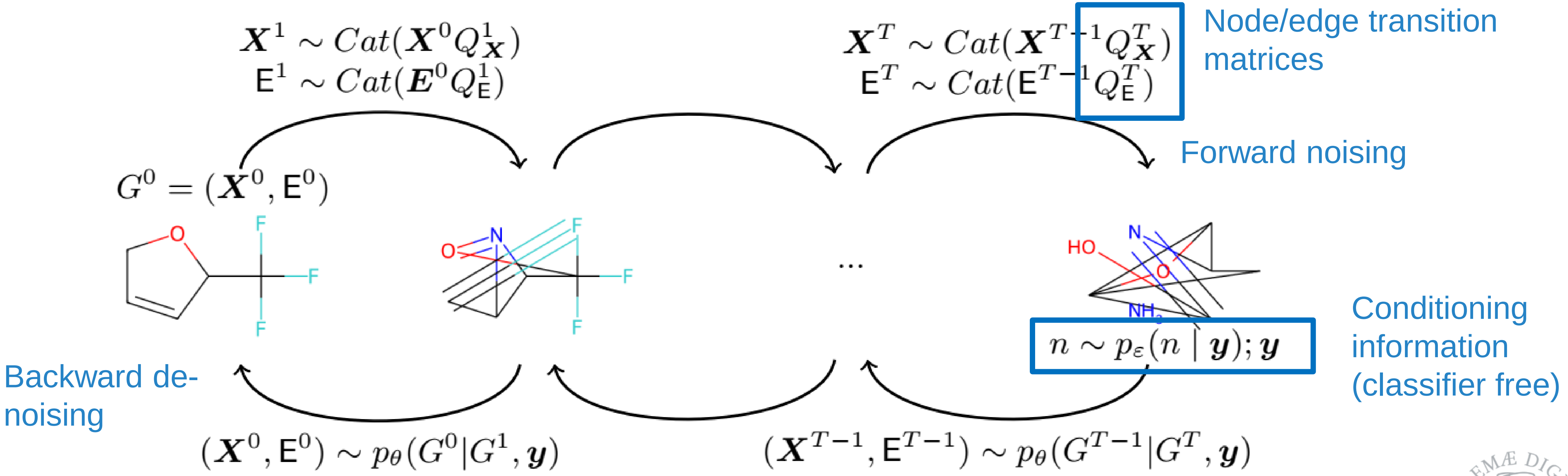


✓Order is **deterministic** and the molecule can be reconstructed

✓Keep a **vocabulary of all possible fragments** found in a dataset

✓Graphs are transformed into **fragment sequences**

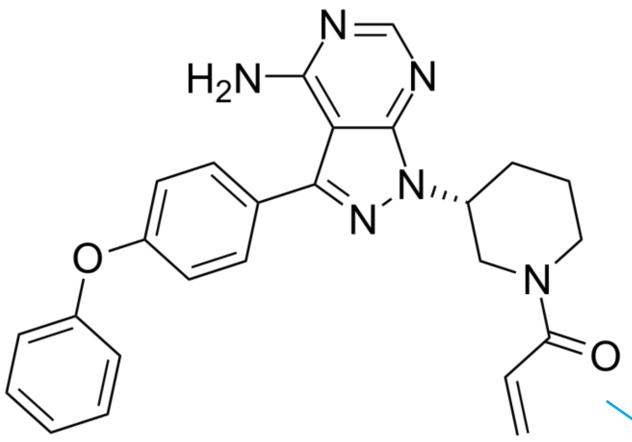
Diffusion Models for Graphs

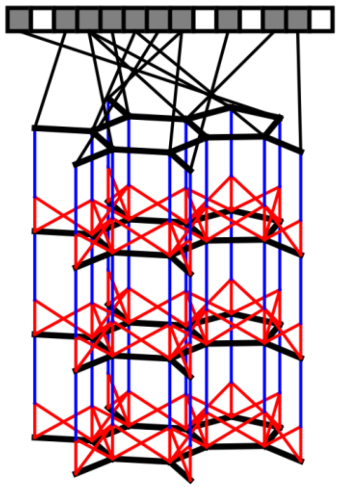


Ninniri, Podda, Bacciu AAAI-WS 2024

Applications

Predicting Properties of Chemical Compounds





$\sim 10^{-2}$ seconds

Simulation methods

$\sim 10^{3}$ seconds

Toxicity

Solubility

Quantum mechanical properties

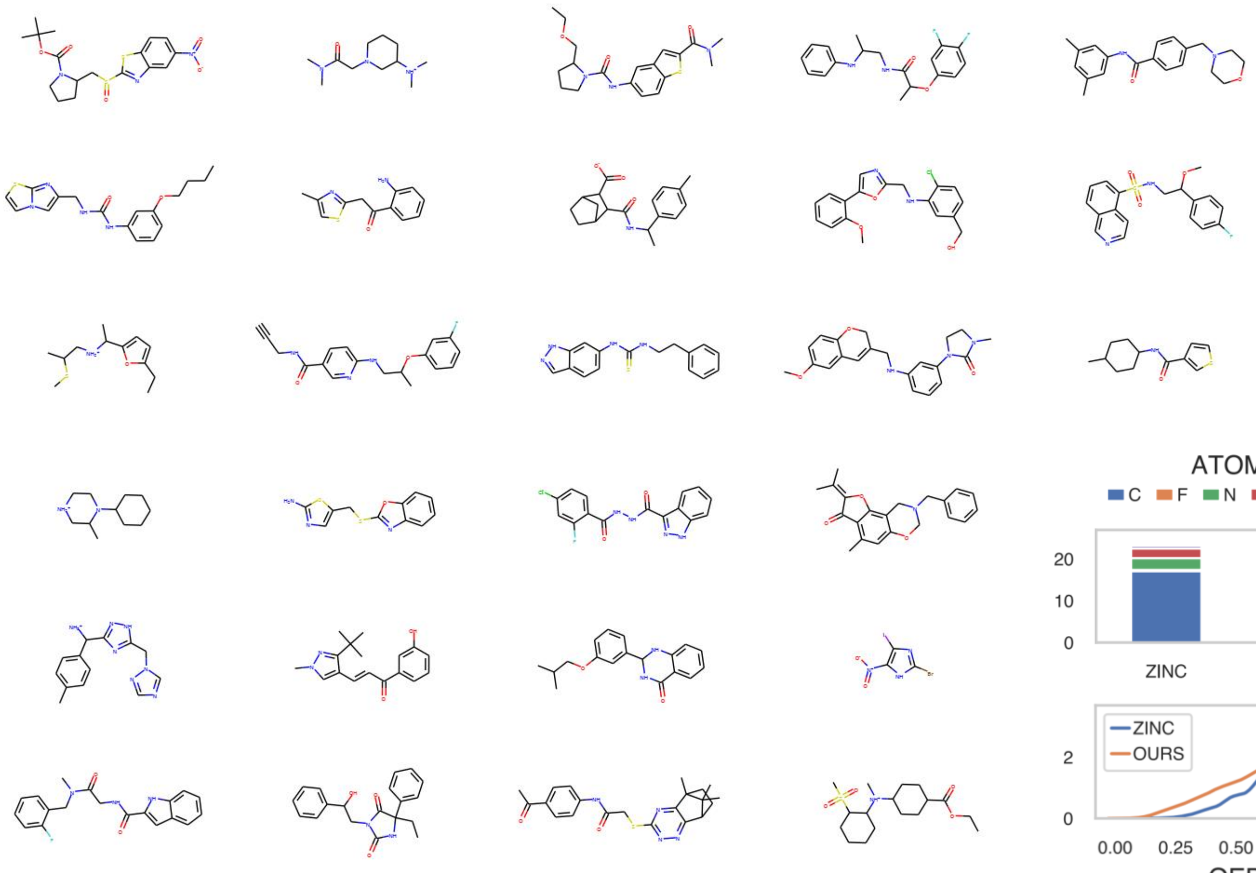
Micheli et al, JCICS 2001

Duvenaud, Maclaurin et al, NIPS 2015

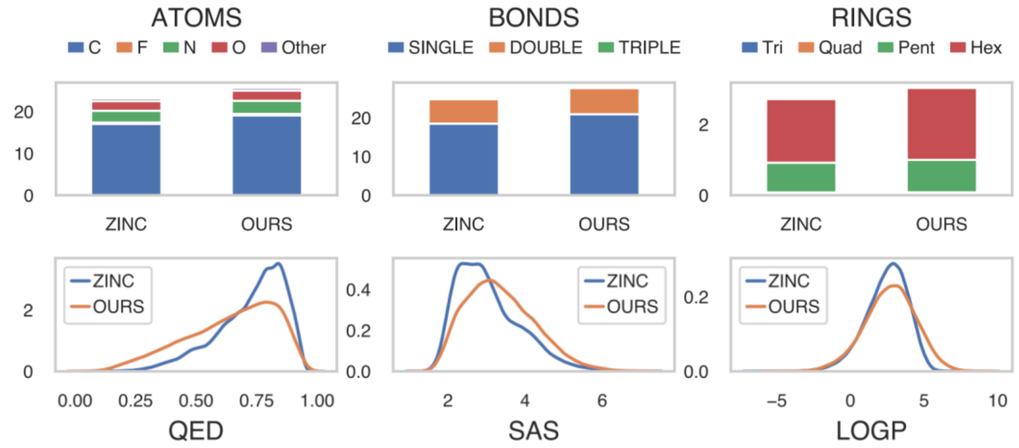
Gilmer et al, ICML 2017

Generating Molecules

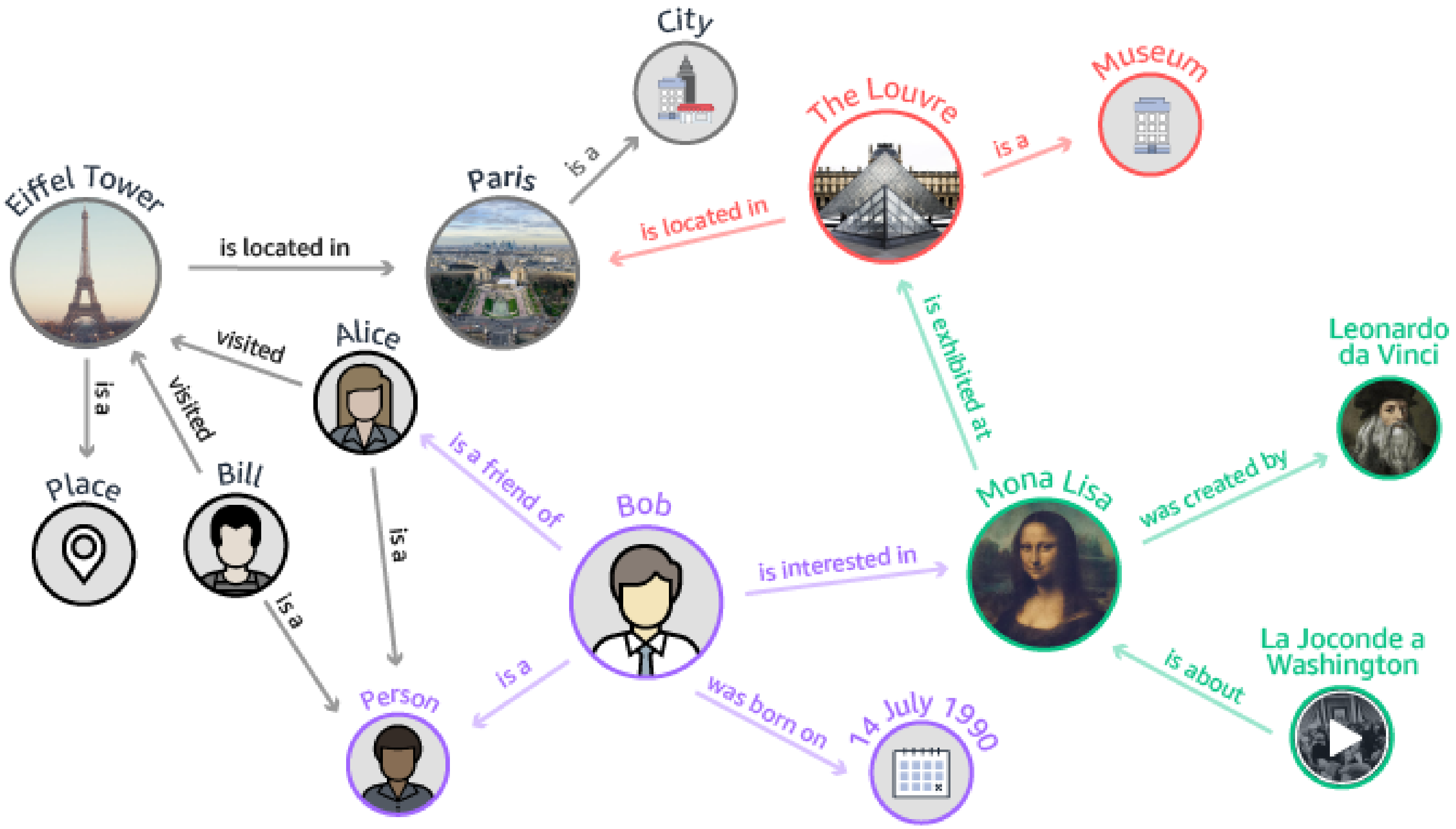
Podda, Bacciu, Micheli, AISTATS 2020



Fragment-based deep molecule generation



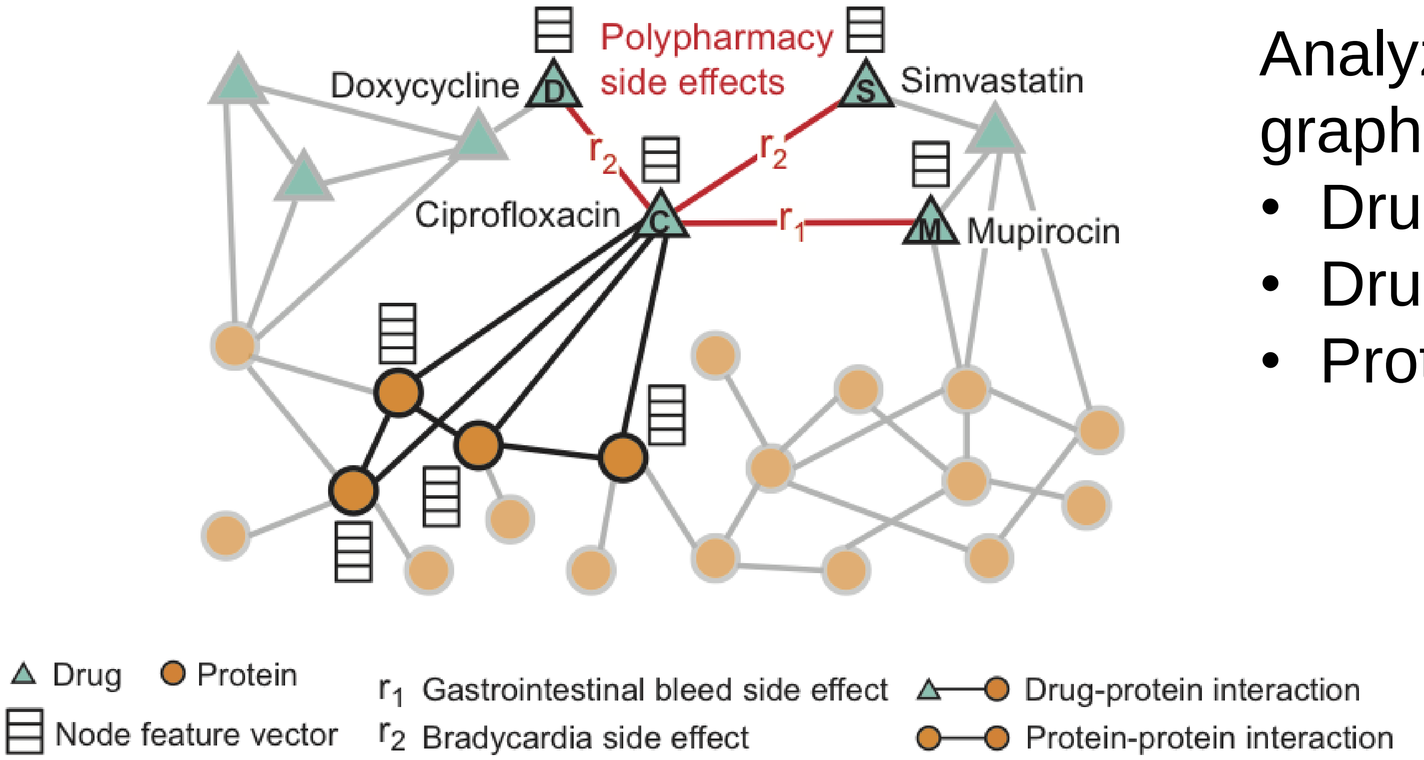
Knowledge graphs



A natural way of representing known entities and relationships in a domain

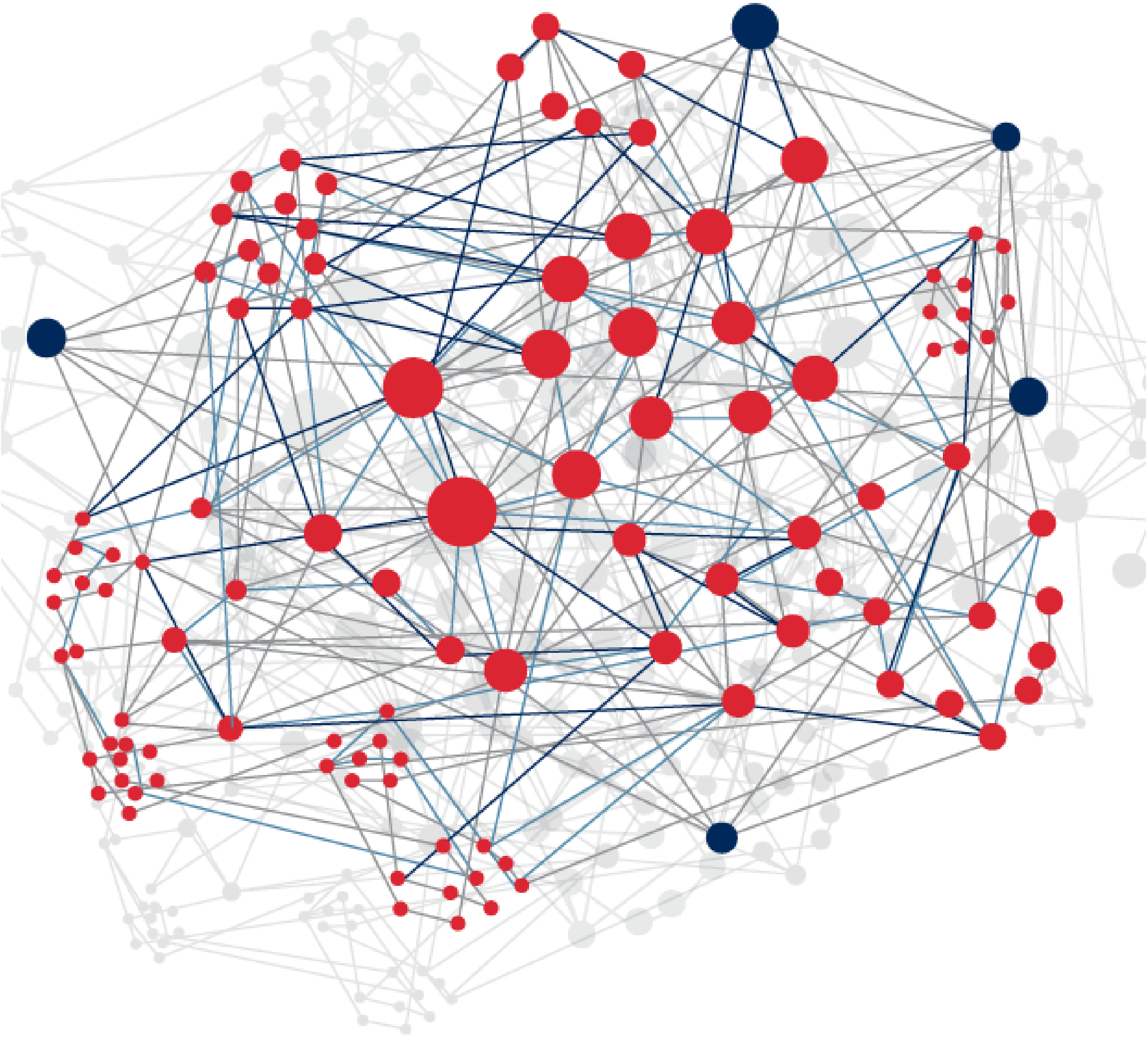
Node/link embeddings are numerical encodings of entities and relationships

Side Effects of Drug Combinations



Analyzing a multimodal graph of interactions
- Drug-drug
- Drug-protein
- Protein-protein

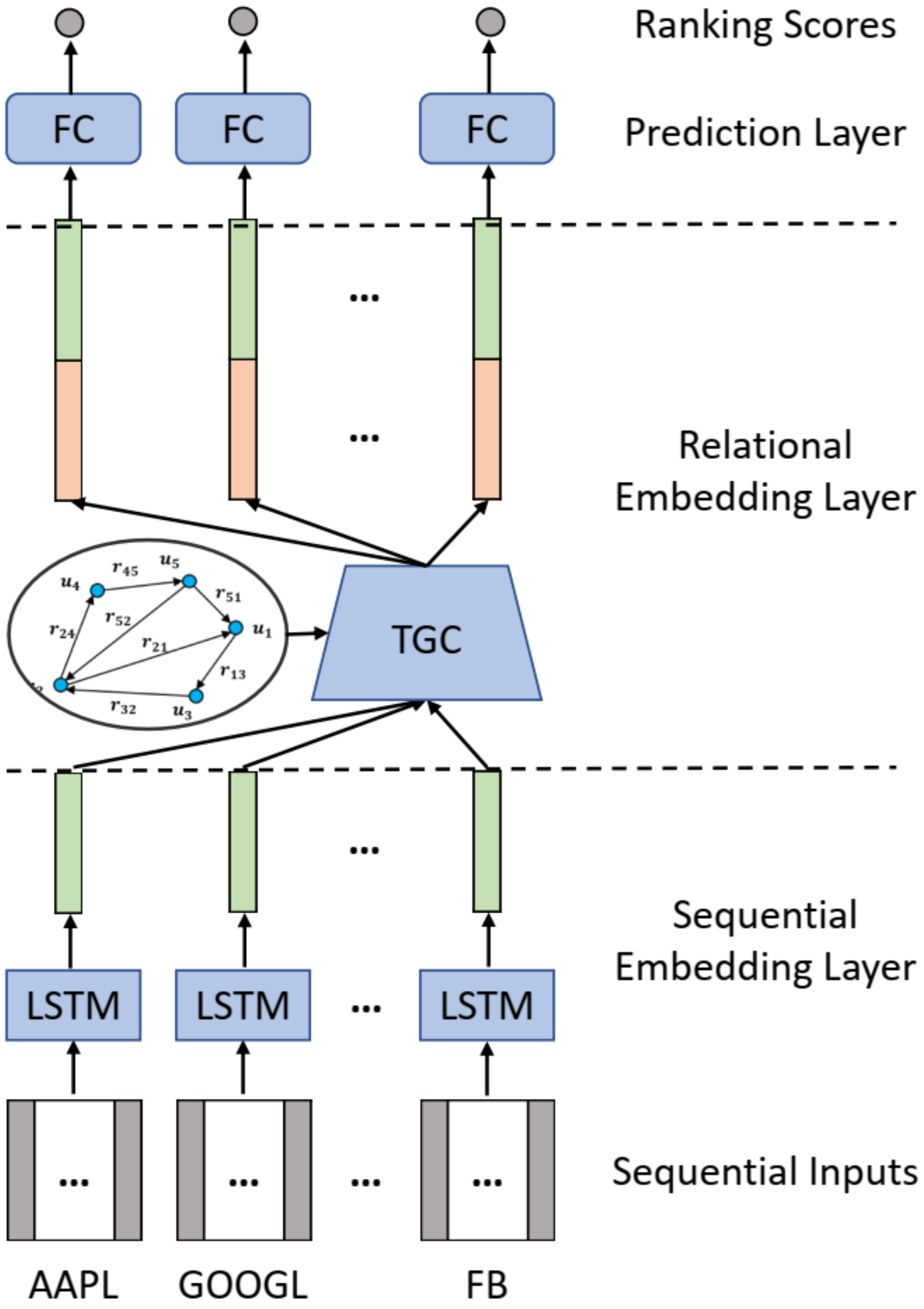
Zitnik, Agrawal, Leskovec, Bioinformatics 2018



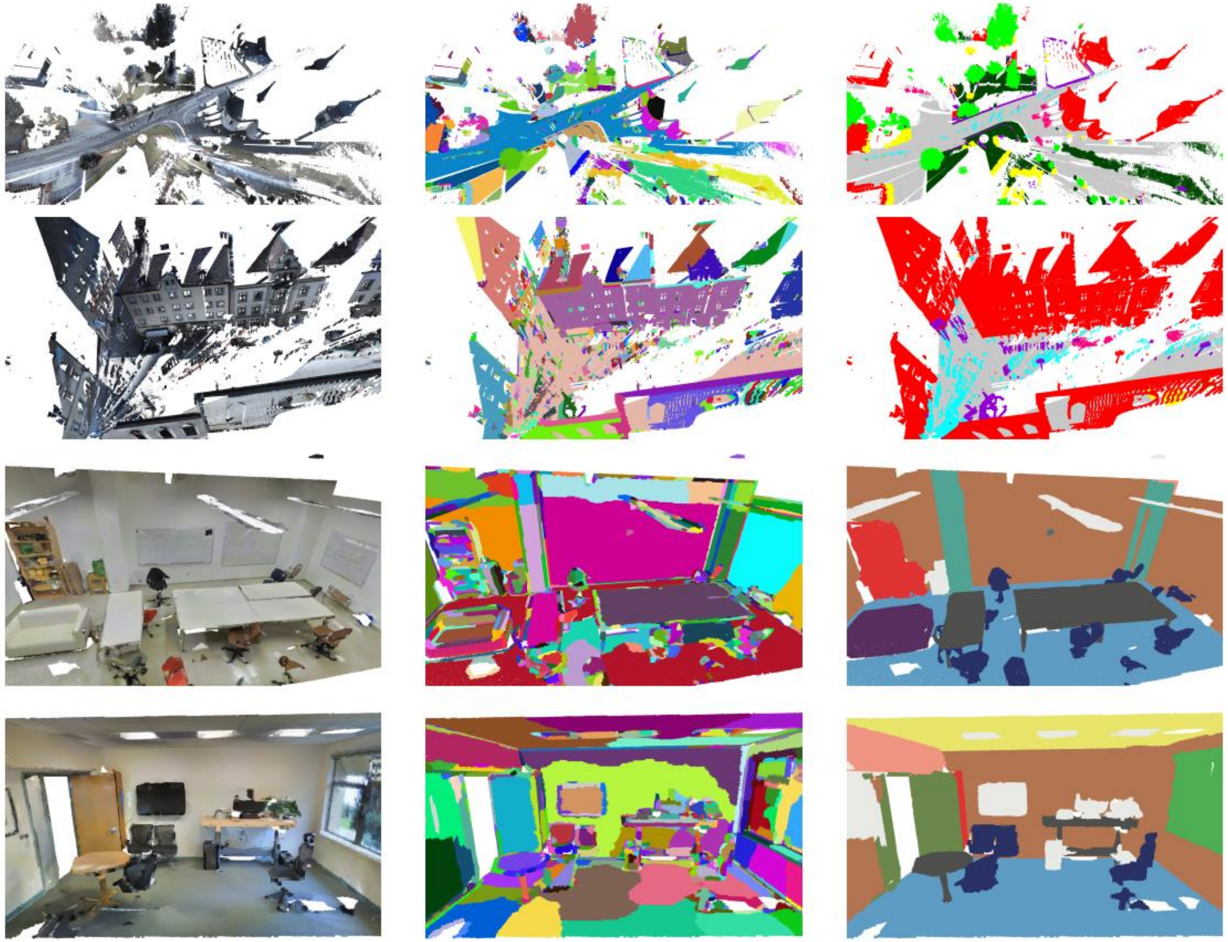
Recommendation Systems

…and other kinds of social network analyses

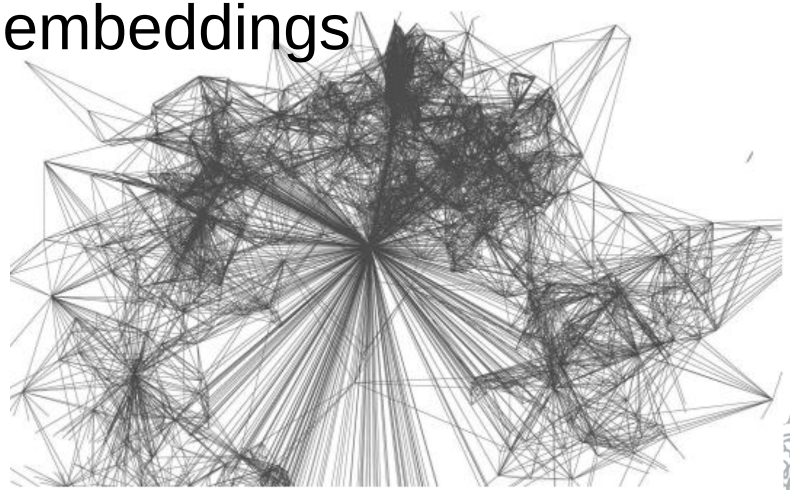
Relational Stock Learning



Point Clouds – Semantic Segmentation

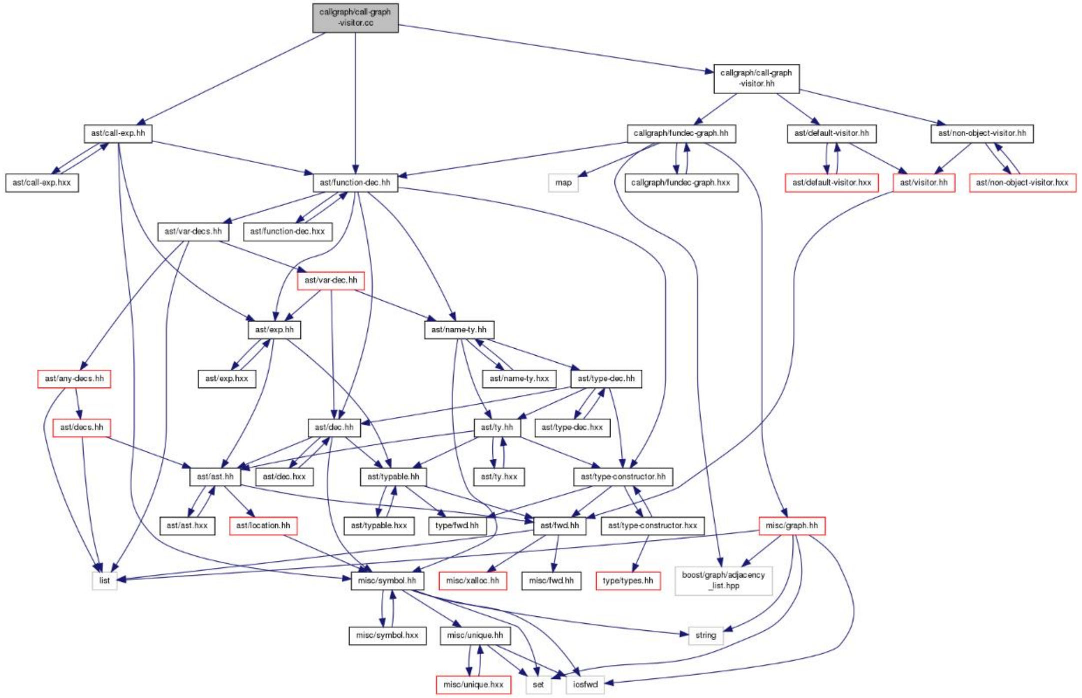


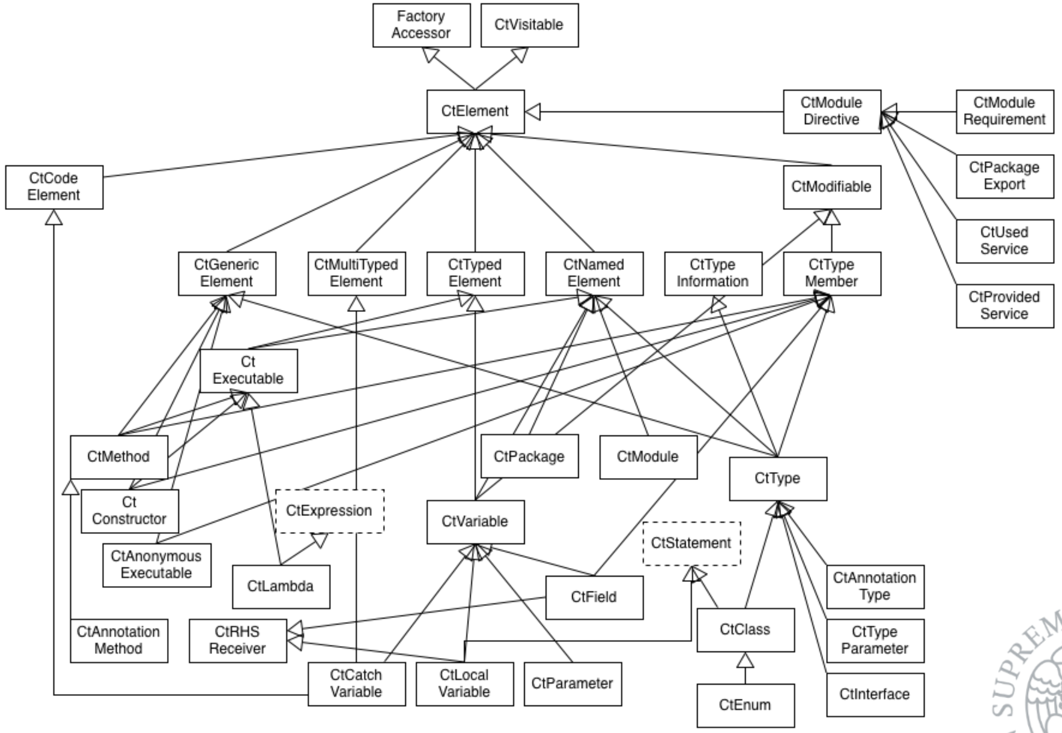
Build point cloud graphs and train semantic class predictors based on vertex embeddings



Landrieu, Simonovsky, CVPR 2018

Analysis of ICT systems/Blockchains

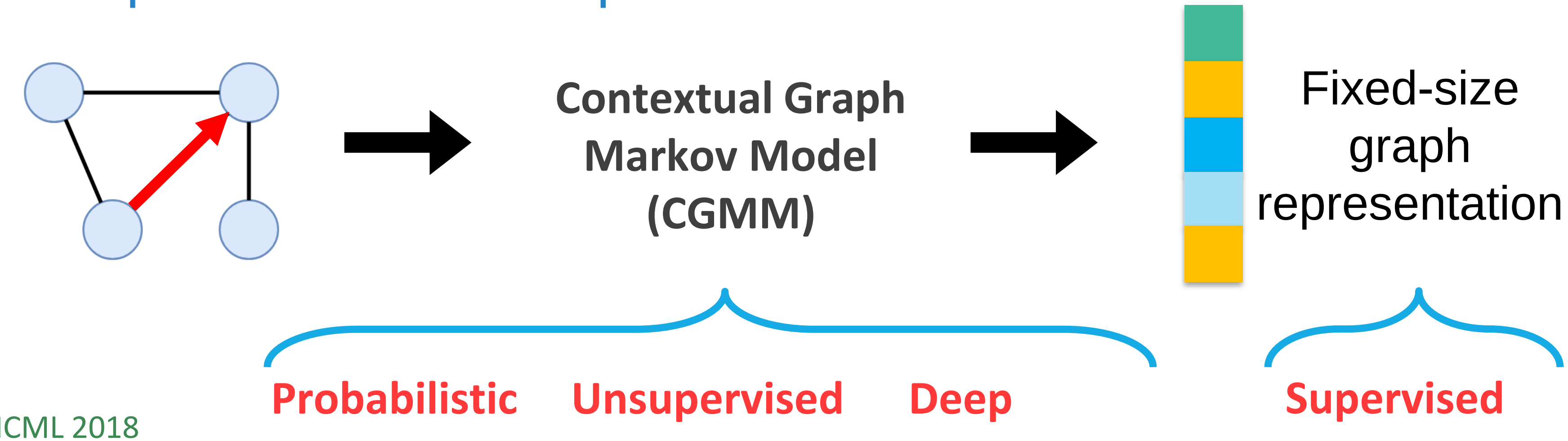




Advanced Topics

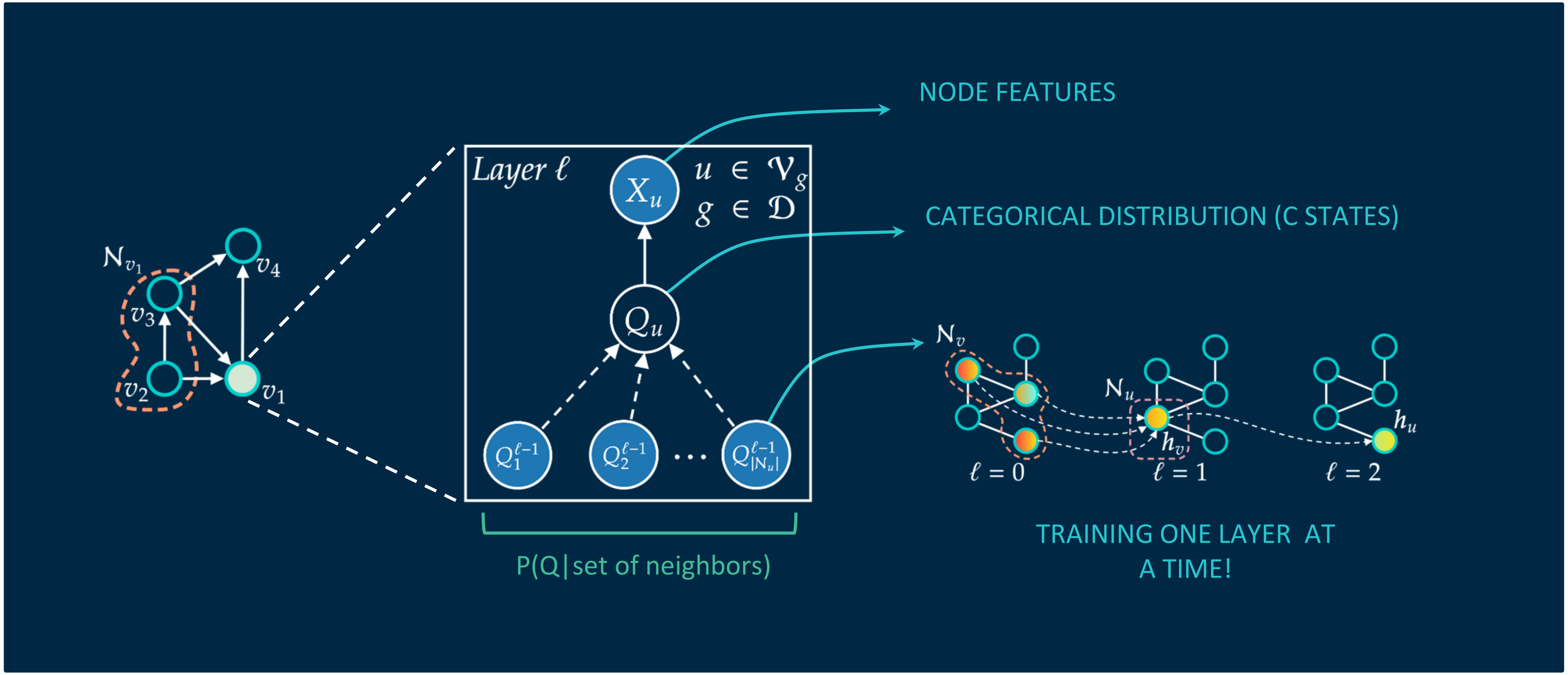
Unsupervised Graph Embeddings

- Learn unsupervised node and graph embeddings
  - Requiring less supervised labelling
  - Reusing embeddings across multiple tasks
- Mix supervised and unsupervised modules



Bacciu, Errica, Micheli , ICML 2018

Contextual Graph Markov Model (CGMM)



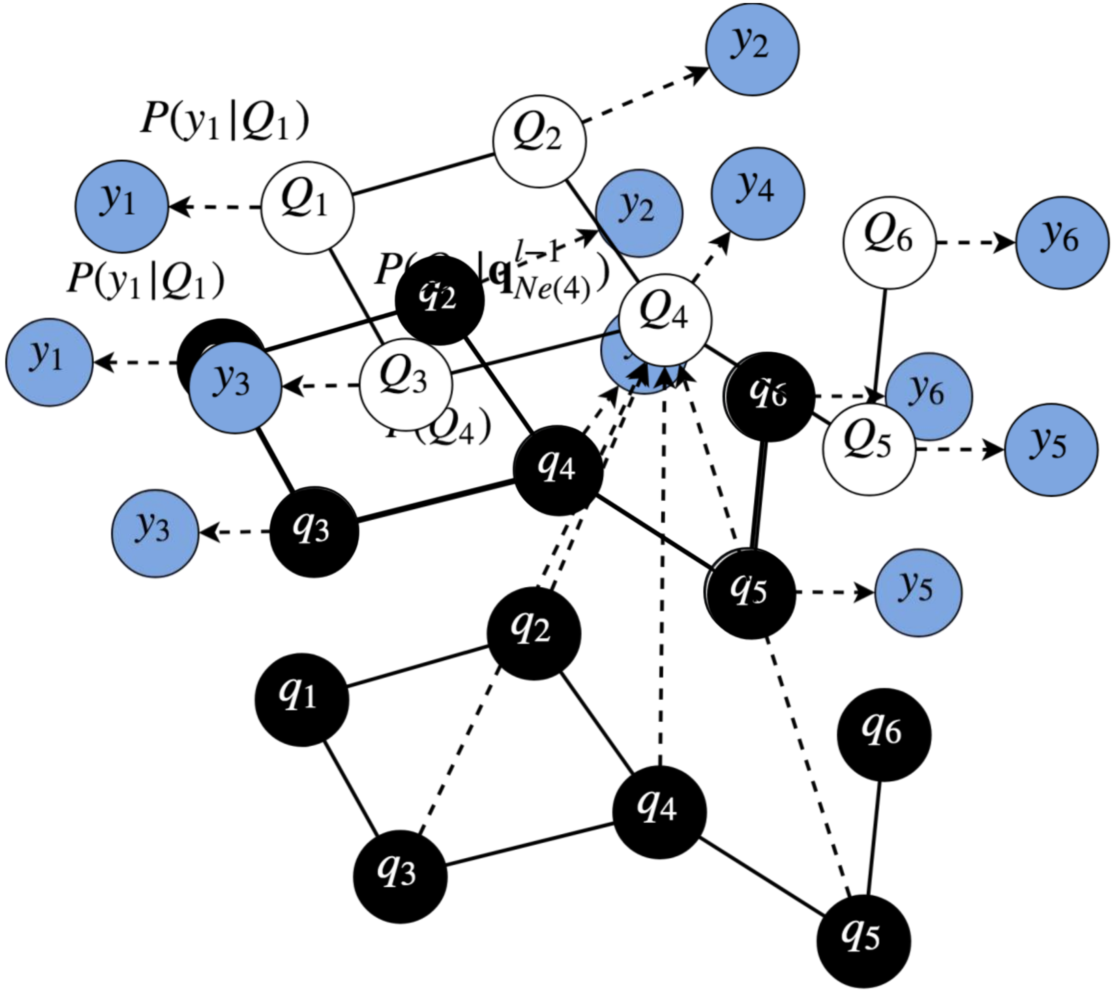
Incremental Construction

1. Map the graph to the model (base case)

2. Perform inference and freeze states

3. Add a new layer and use frozen states as observed variables in the graphical model

Go back to step 2



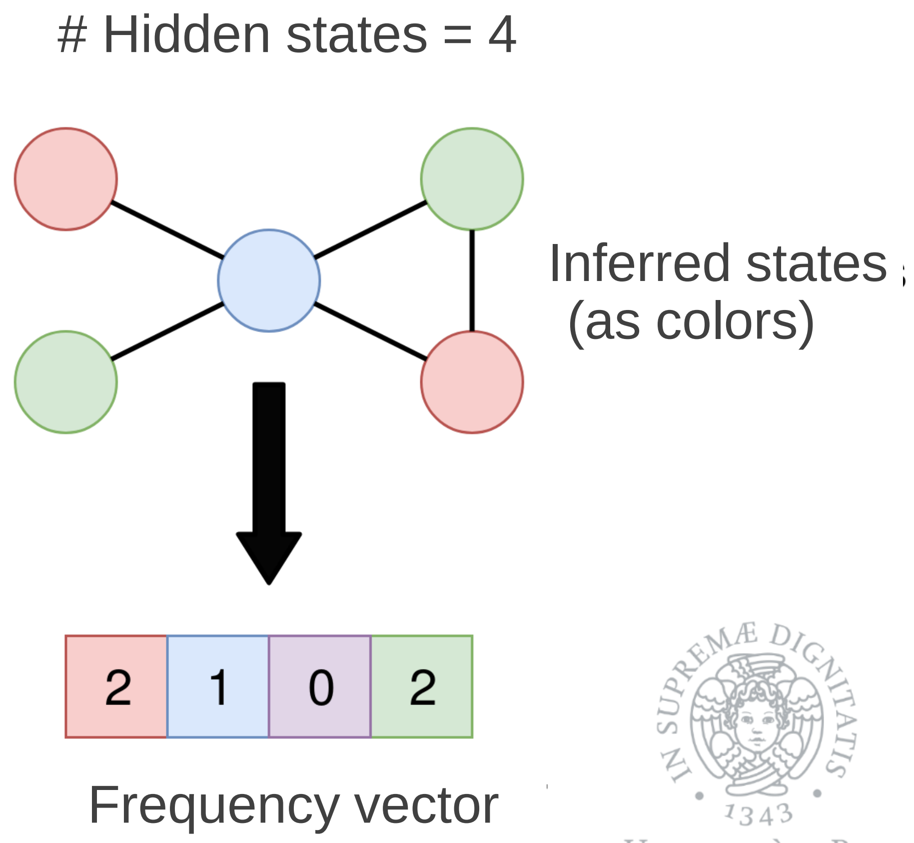
Computing embedding

✓Finding the most likely **state assignment**

$$\max_i P(y_u | Q_u = i) P(Q_u = i | \mathbf{q}_{\mathcal{N}(u)})$$

✓The inferred latent states are used as observable variables in subsequent layers

✓A **fixed-size vector of states frequencies** as graph encoding



CGMM Layer Training

A maximum likelihood approach to learning

$$\mathcal{L} = \prod_{g \in G} \prod_{u \in g} \sum_{i}^{C} P^l(y_u | Q_u = i) P^l(Q_u = i | \underbrace{\mathbf{q}_{\mathcal{N}(\mathbf{u})}^{\mathbf{L_{prec}}}(\mathbf{g})})$$

Assumption: i.i.d. graphs

Emission distrib.

Switching Parents distrib.

Transition distrib.

Split by layer

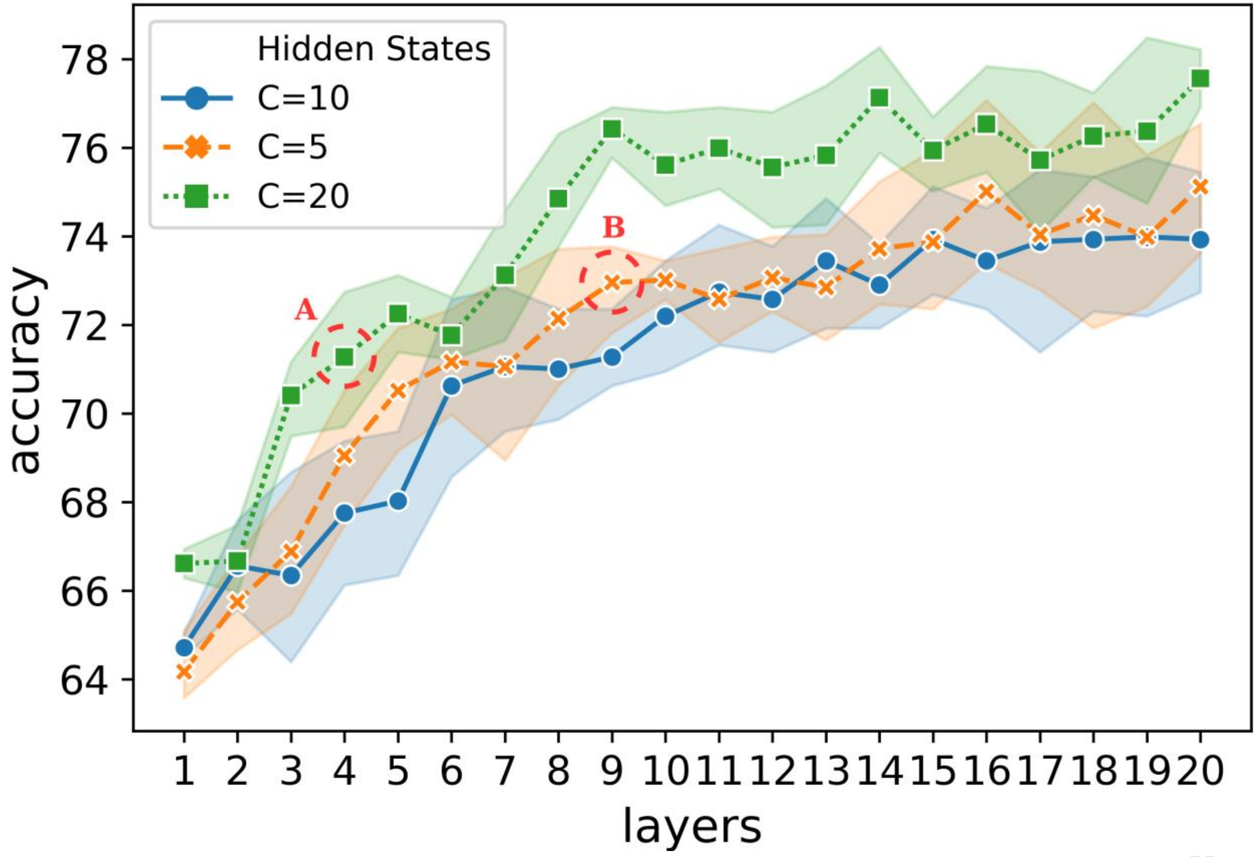
and by arc

$$= \prod_{g \in G} \prod_{u \in g} \sum_{i}^{C} P^l(y_u | Q_u = i) \sum_{\bar{l} \in \bar{L}} P(L_u = \bar{l}) \sum_{a=1}^{A} P^{\bar{l}}(S_u = a) \frac{1}{|\mathcal{N}^{\bar{l}a}(u)|} \sum_{j}^{C} P^{\bar{l}a}(Q_u = i | Q_*^{\bar{l}a} = j) \sum_{v \in \mathcal{N}^{\bar{l}a}(u)} q_v^{\bar{l}}(j)$$

Trained by Expectation-Maximization

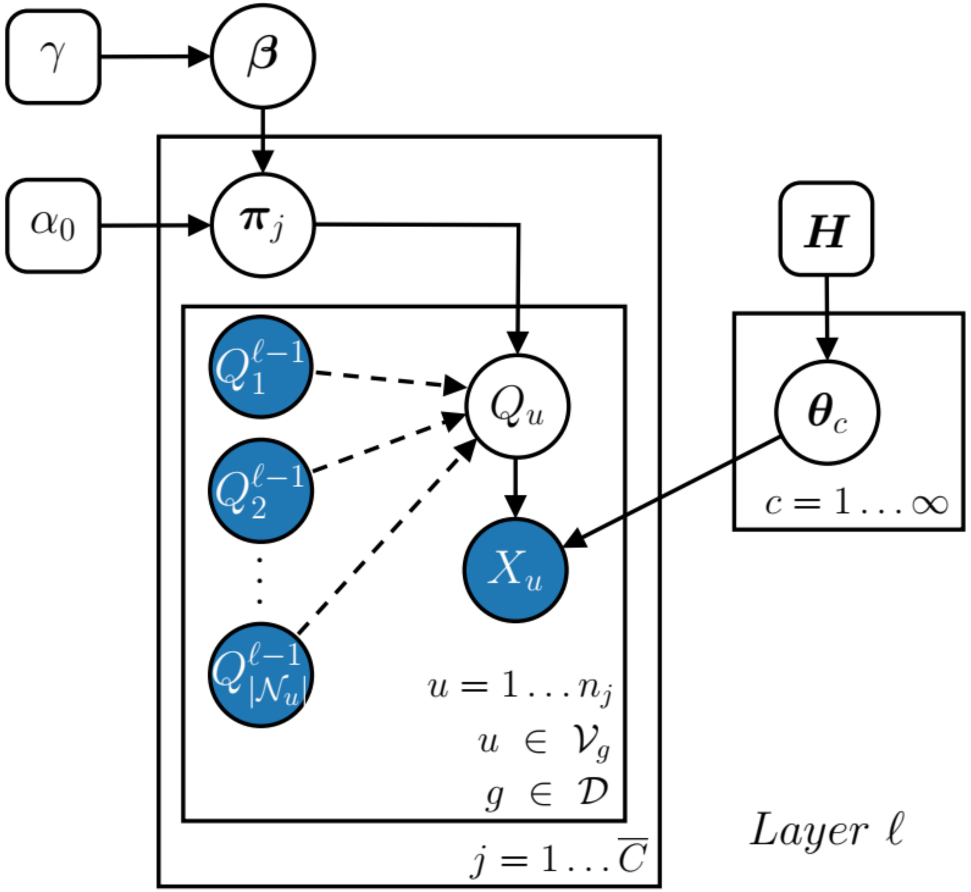
CGMM – Depth Matters…

…possibly more than width



Bacciu, Errica, Micheli , JMLR 2020

To infinity and beyond



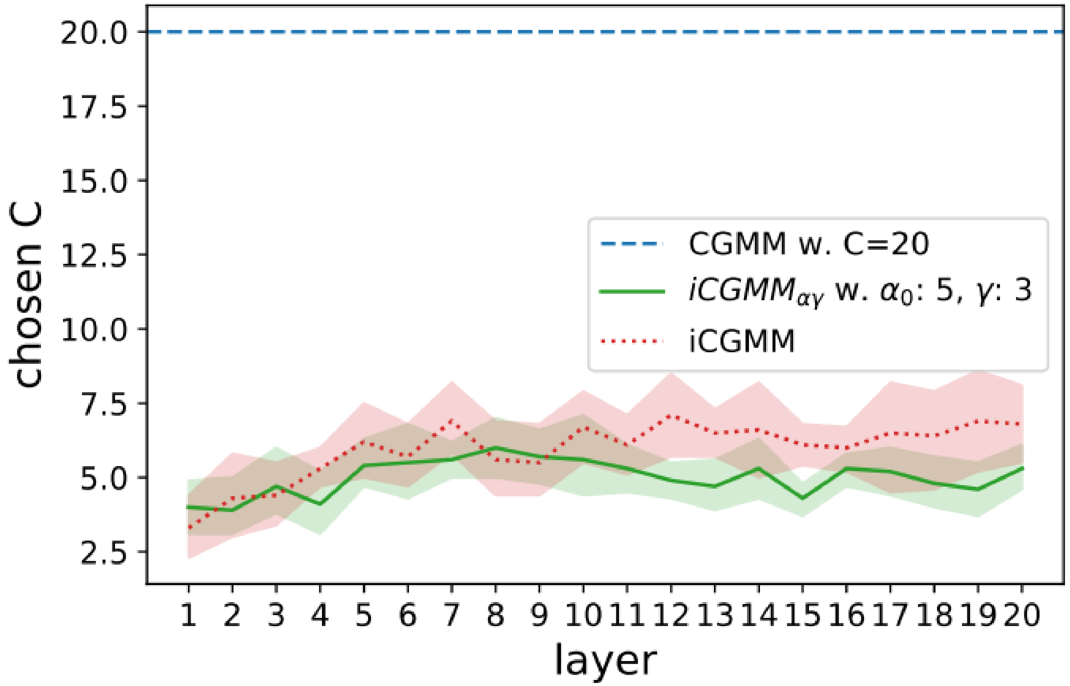
The Infinite CGMM

- ✓ Hierarchical Dirichlet process to sample (potentially) infinitely many hidden states
- ✓ Automatically learn the size of node embedding space from data
- ✓ Choice of observations' groups determined by neighbors' states
- ✓ Batch version for larger datasets

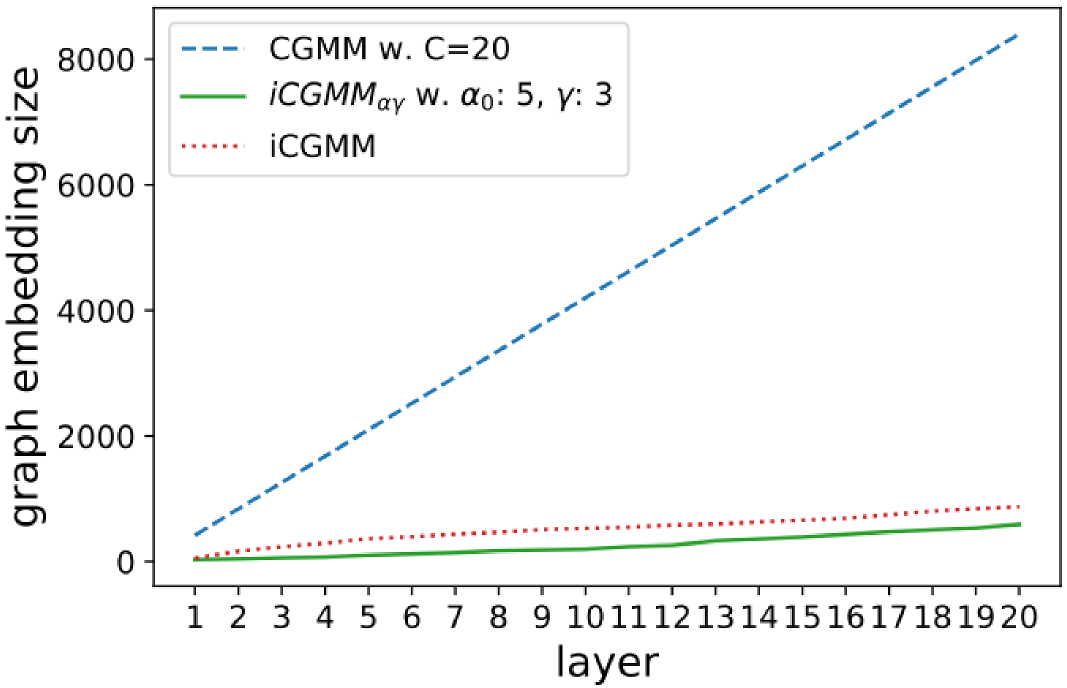
Castellana, Errica, Bacciu, Micheli , ICML 2022

ICGMM – Finer grained control on hidden space

CHOSEN STATES PER LAYER

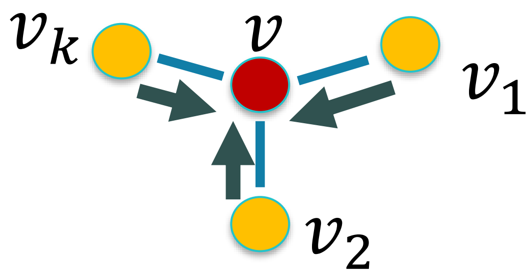


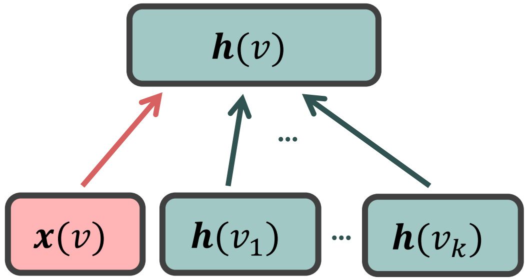
CUMULATIVE GRAPH EMBEDDING SIZE



UNIVERSITÀ DI PISA

Graph embedding by learning-free neurons





- Each vertex in an input graph is encoded by the hidden layer

input weight matrix

hidden weight matrix

$$\boxed{\mathbf{h}(v)} = \tanh(\boxed{\mathbf{V}}\,\boxed{\mathbf{x}(v)} + \sum_{v'\in N(v)} \boxed{\mathbf{W}}\,\boxed{\mathbf{h}(v')})$$

embedding (state) of vertex v

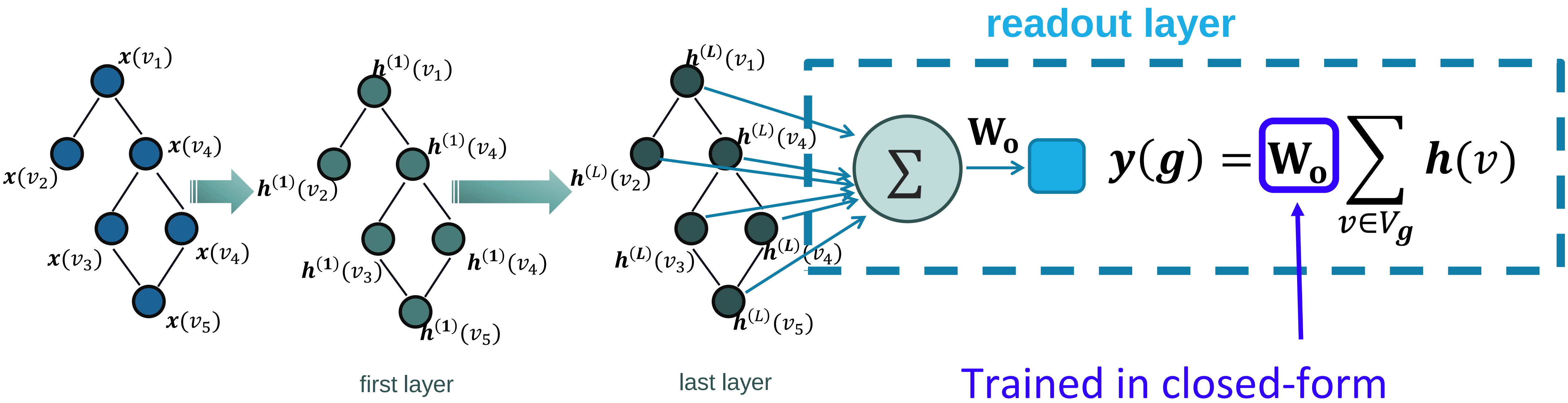
input feature of vertex v

embedding (state) of neighbors of vertex v

Need this to be contractive to ensure convergence of embedding

Deep Reservoirs for Graphs

Gallicchio & Micheli. *AAAI* 2020.

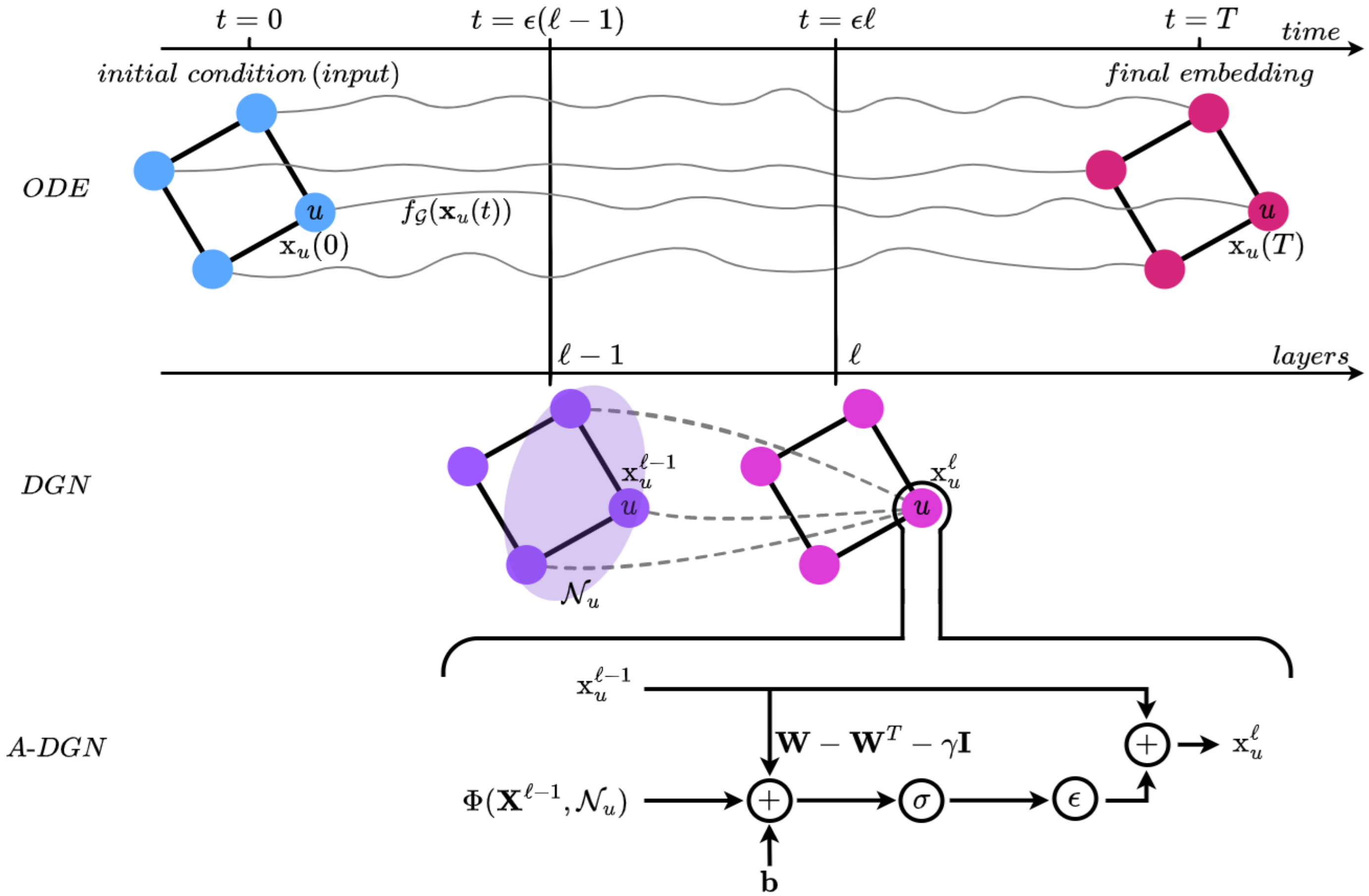


$$y(g) = \boxed{\mathbf{W_o}} \sum_{v \in V_g} h(v)$$

Deep reservoir embedding

Trained in closed-form (e.g., pseudo-inversion, ridge regression)

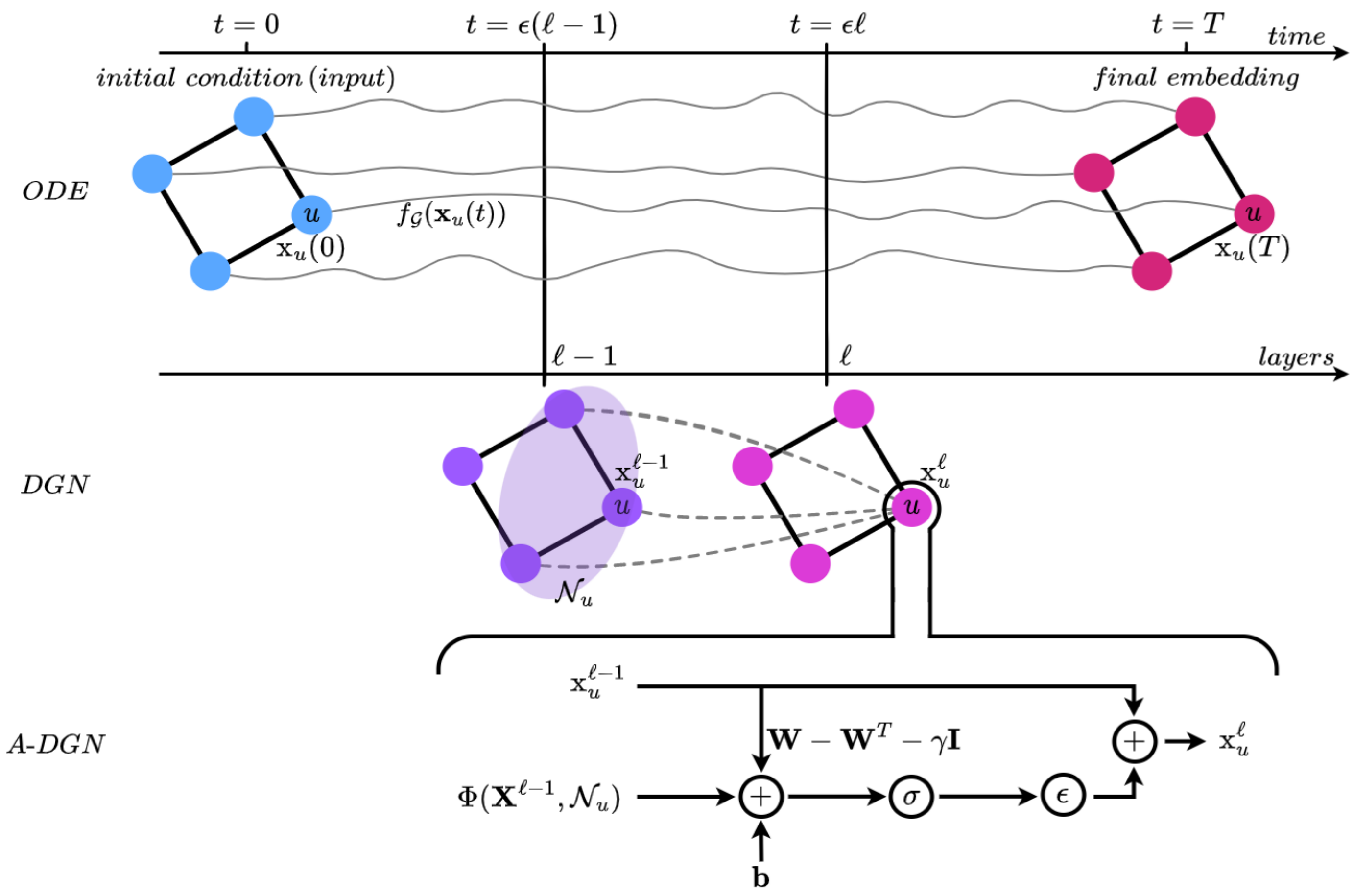
A Dynamical Systems View on Deep Graph Networks



- Node message passing can also be seen as a discretization of a continuous dynamical process

$$\begin{cases} \frac{\partial \mathbf{x}_u(t)}{\partial t} = f_{\mathcal{G}}(\mathbf{x}_u(t)) & t \in [0, T], \\ \mathbf{x}_u(0) = \mathbf{x}_u^0 \in \mathbb{R}^d \end{cases}$$

A Dynamical Systems View on Deep Graph Networks



- Node message passing can also be seen as a discretization of a continuous dynamical process
- The graph neural network has as many layers as the length of the unfolded ODE
- Neural (Graph) ODE

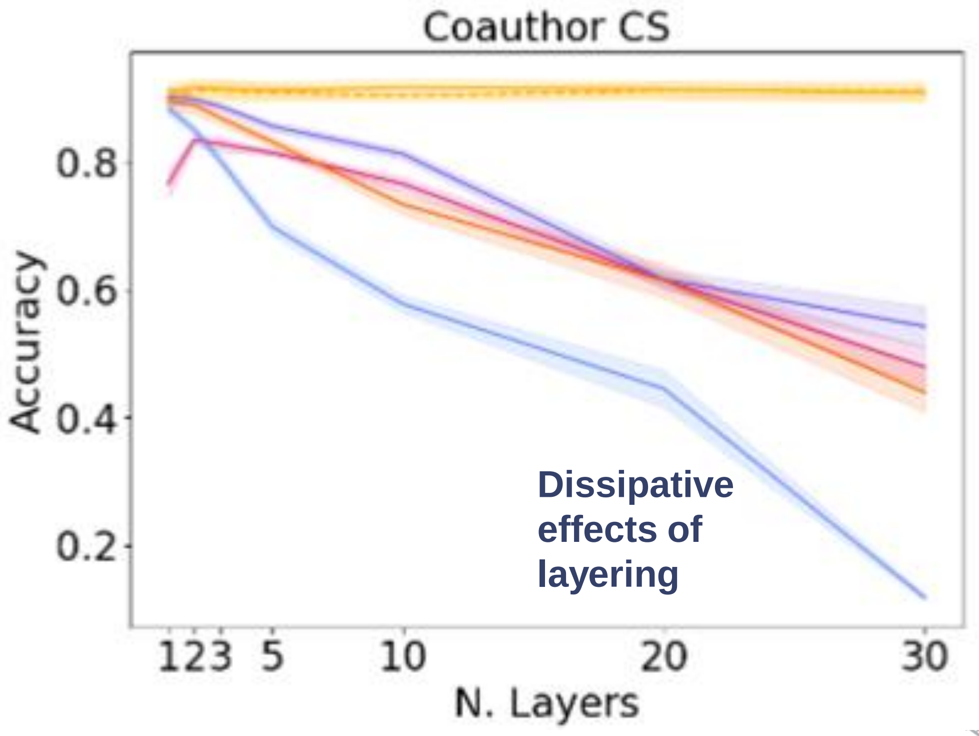
Non-Dissipative Propagation

- An intermediate step is fundamental before working with dynamic graphs to obtain a stable and non-dissipative message passing
- The primary challenge in the graph representation learning is capturing and encoding structural information in the learning model
- Exploiting local interactions between nodes might not be enough to learn representative embeddings
  - A specific range of node interactions is required to effectively solve the problem
  - The DGN requires a specific number (possibly large) of layers
  - Over-squashing problem

Non-Dissipative Message Passing

Gravina, Bacciu, Gallicchio, ICLR 2023

- Many-layer networks are needed to capture long range node interactions into representative embeddings
- Leverage the ODE formulation of DGNs to optimize forward and backward message propagation



Non-Dissipation by Anti-Symmetric Parameterization

Forward Euler discretization of Graph ODE

$$\mathbf{x}_u^{\ell} = \mathbf{x}_u^{\ell-1} + \epsilon\sigma\left((\mathbf{W} - \mathbf{W}^T - \gamma\mathbf{I})\mathbf{x}_u^{\ell-1} + \Phi(\mathbf{X}^{\ell-1}, \mathcal{N}_u) + \mathbf{b}\right)$$

Step size

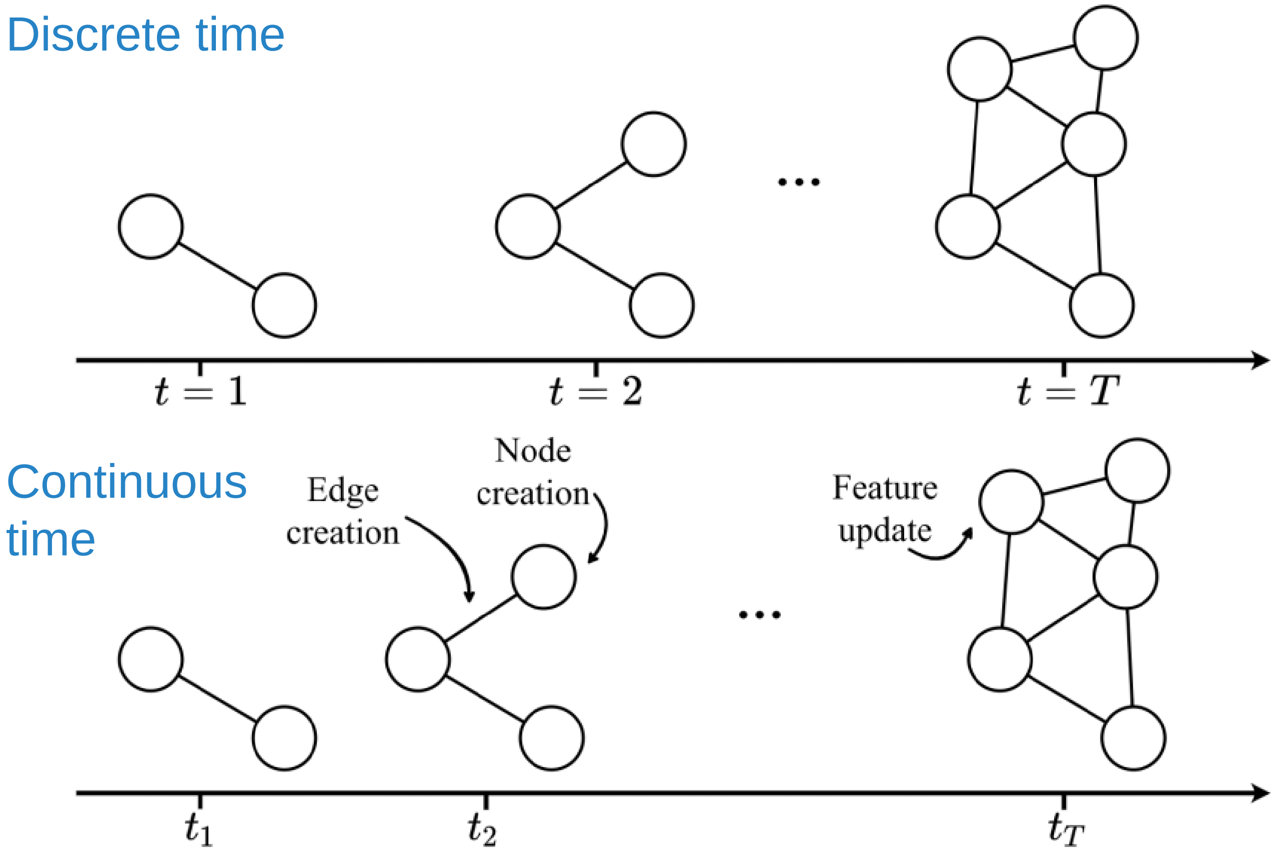
Monotonically non-decreasing activation function, e.g., tanh, relu

Anti-symmetric weight matrix allowing stable and non-dissipative behavior of the ODE (eigenvalues of the Jacobian are all imaginary)

Diffusion term that preserves the stability of the discretized system

Neighborhood aggregator (any standard DGNs)

Learning with Dynamic Graphs



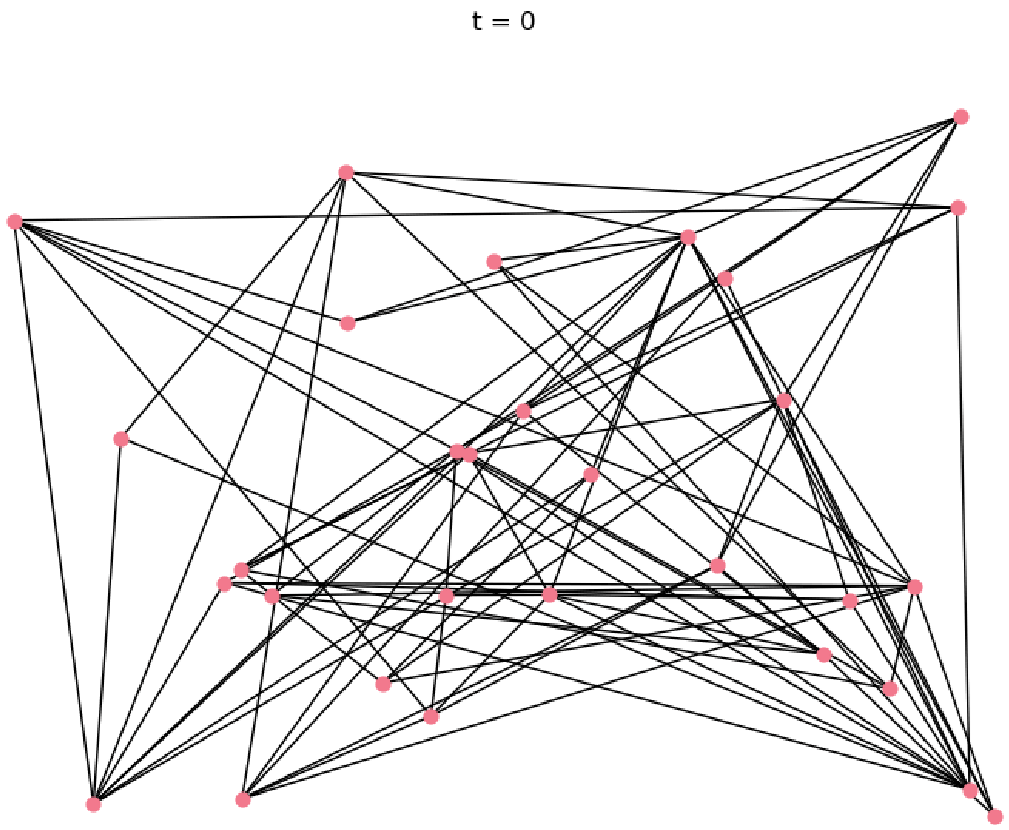
Graphs evolve with time in feature, connectivity and topology

- Spatio-temporal networks
- Graph streams

R. Trivedi et al ICRL 2019

Gravina & Bacciu, TNNLS 2024 (Survey)

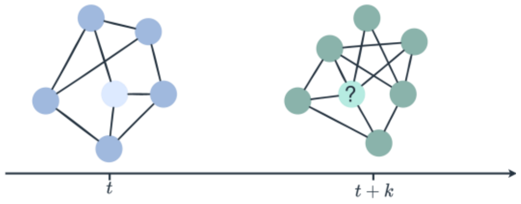
Dynamic Graphs Vs Static DGNs



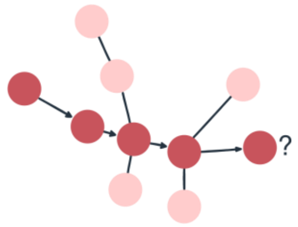
- DGNs cannot be directly applied to all real-life graphs
  - Most real-life graphs are dynamic
  - Majority of DGN approaches assume that the input graph is static
- Ignoring temporal information can make the problem impossible to solve
- **Objective**: develop methods that are able to exploit both spatial and temporal information

Common Tasks with Dynamic Graphs

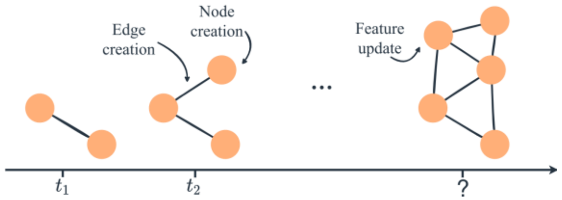
- Future link/node prediction
  - Predict at time t + k



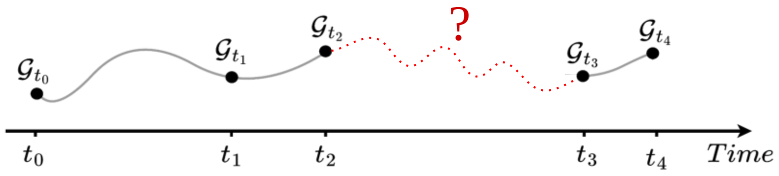
- Path classification
  - E.g. predict path congestion



- Event time prediction
  - When an event will occur?



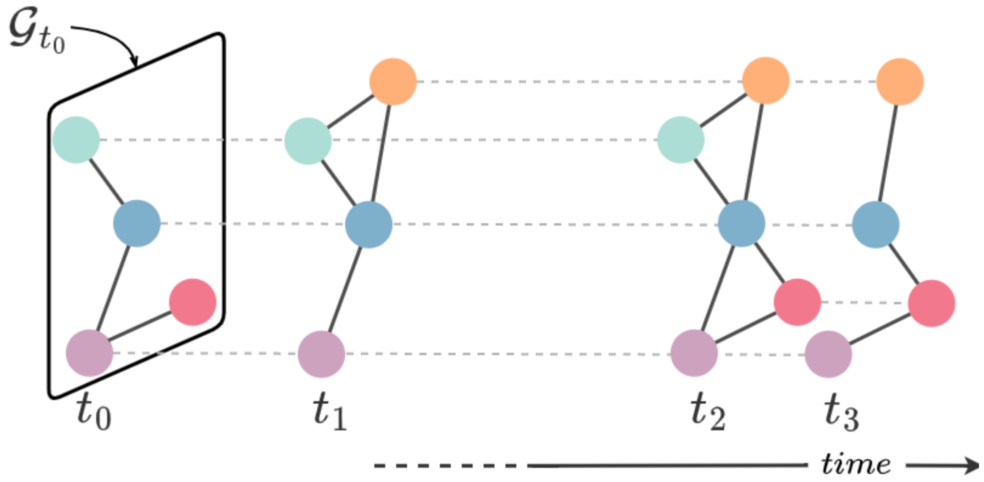
- Imputation



UNIVERSITÀ DI PISA

Irregular Graph Streams

Gravina et al, IJCAI 2024



- Graphs occur in the stream as irregular events
- Can be treated naturally by a neural ODE that propagates node signals between even occurrences

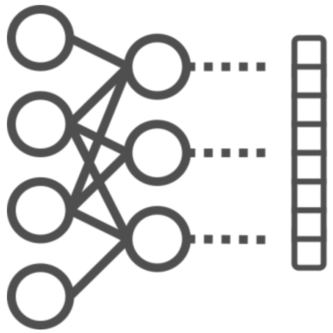
$$\begin{cases} \frac{d\mathbf{x}_u}{dt} = \overbrace{\phi_U}^{\text{Update}} \left(\mathbf{x}_u, \mathbf{z}, \overbrace{\rho}^{\text{Aggregate}} \left(\{\overbrace{\phi_M}^{\text{Message}}(\mathbf{x}_u, \mathbf{x}_v, \mathbf{e}_{vu})\}_{v\in\mathcal{N}_u}\right)\right) \\ \mathbf{x}_u(0) = \underbrace{\overline{\mathbf{x}}_{t_{i-1},u}}_{\text{Observed snapshot of } u \text{ at time } t_{i-1}} \end{cases}$$

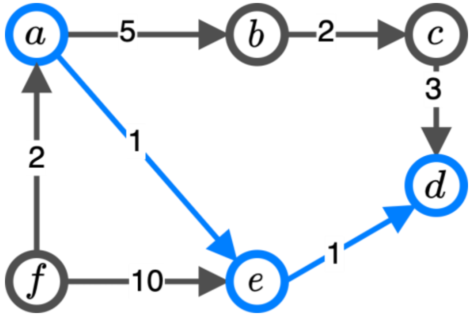
$$\mathbf{x}_u^{\ell+1} = \mathbf{x}_u^{\ell} + \epsilon\phi_U\left(\mathbf{x}_u^{\ell}, \mathbf{z}(t_\ell), \rho\left(\{\phi_M(\mathbf{x}_u^{\ell}, \mathbf{x}_v^{\ell}, \mathbf{e}_{vu}^{\ell})\}_{v\in\mathcal{N}_u(t_\ell)}\right)\right)$$

Can again be solved by forward Euler

Neural Algorithmic Reasoning - Combining algorithms and neural networks

Veličković el al, ICRL 2020





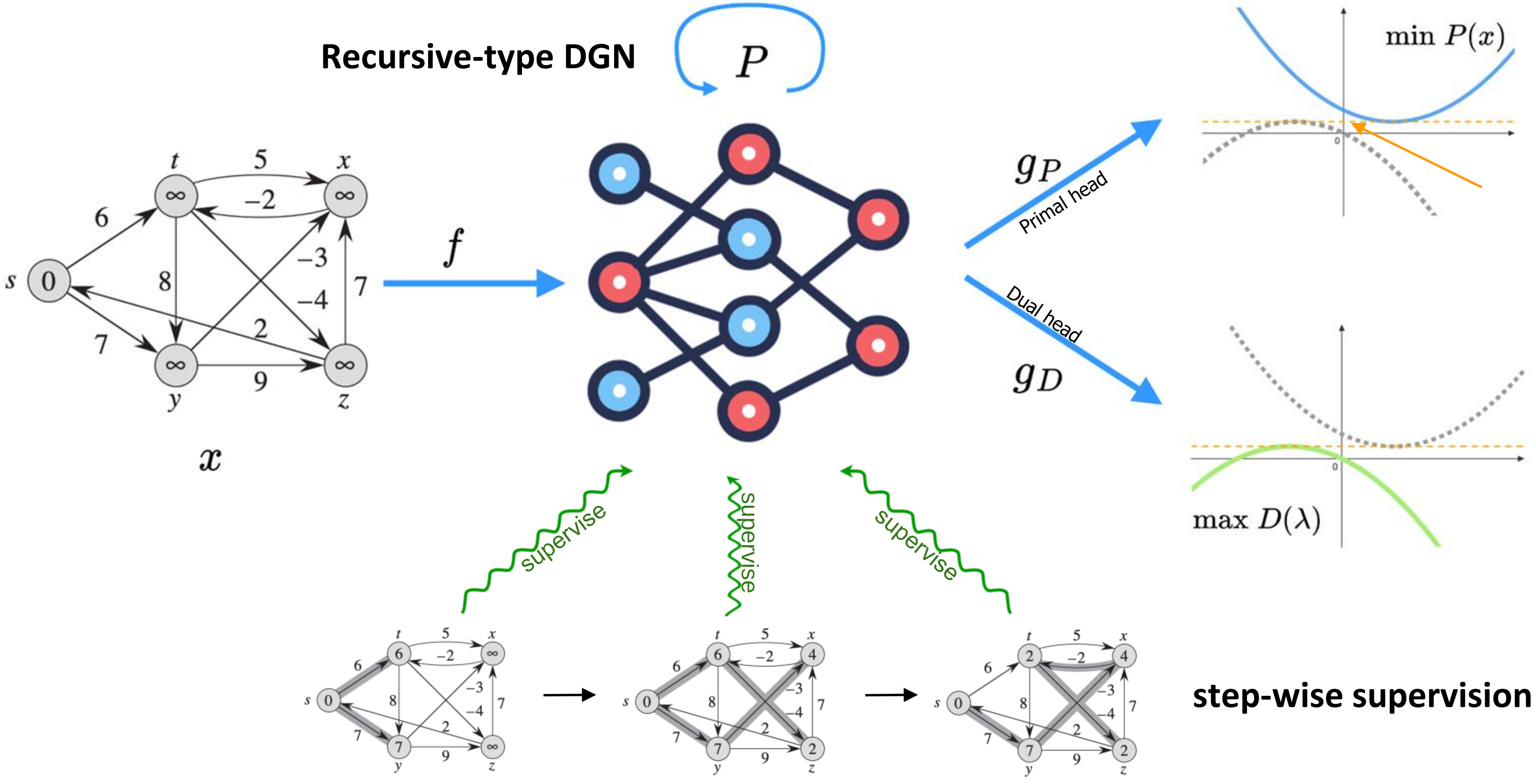
+ Reusable across tasks
+ Executing on noisy conditions
- Sensitive to shift-of-distribution
- No interpretable operations
- Requires lots of data

- Sensitive to task variation
- Input must match pre-conditions
+ Inherent generalisation
+ Interpretable
+ Theoretical guarantees

Can we get the best of both worlds?

IN SUPREMÆ DIGNITATIS 1343 UNIVERSITÀ DI PISA

Learning Algorithmic Reasoning on Graphs



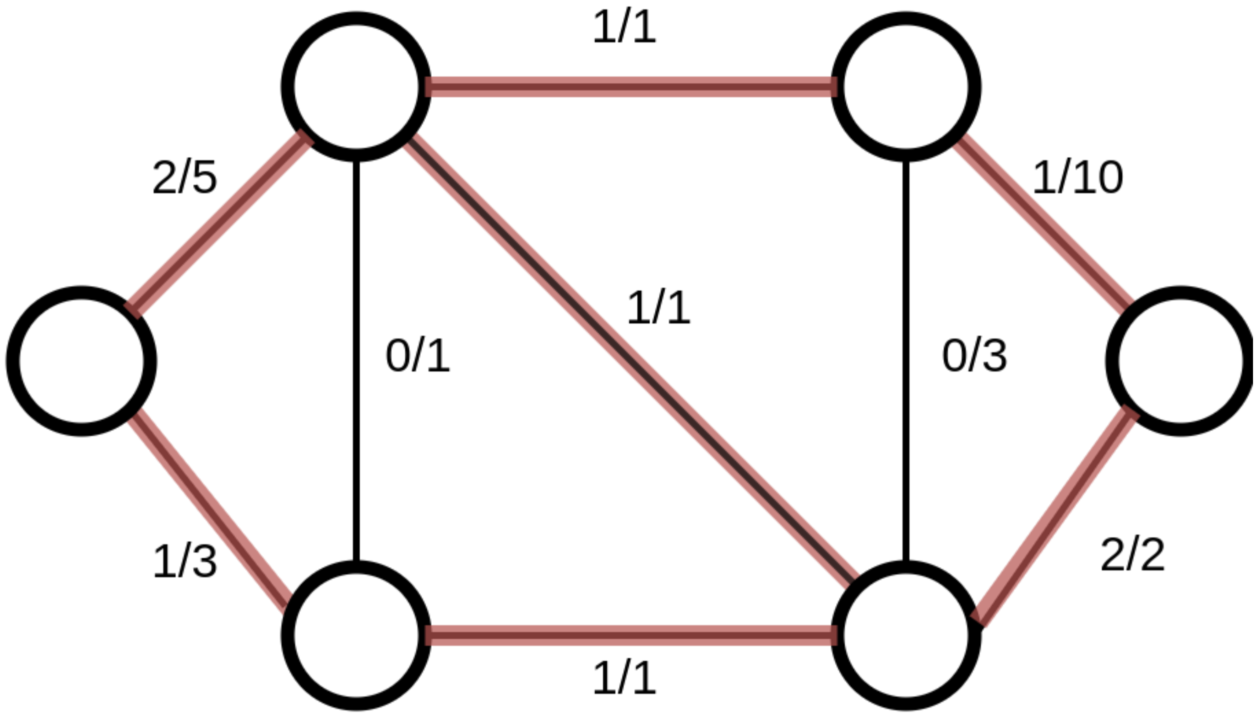
Dual Algorithmic Reasoning

Use knowledge on the structure of the optimization problem

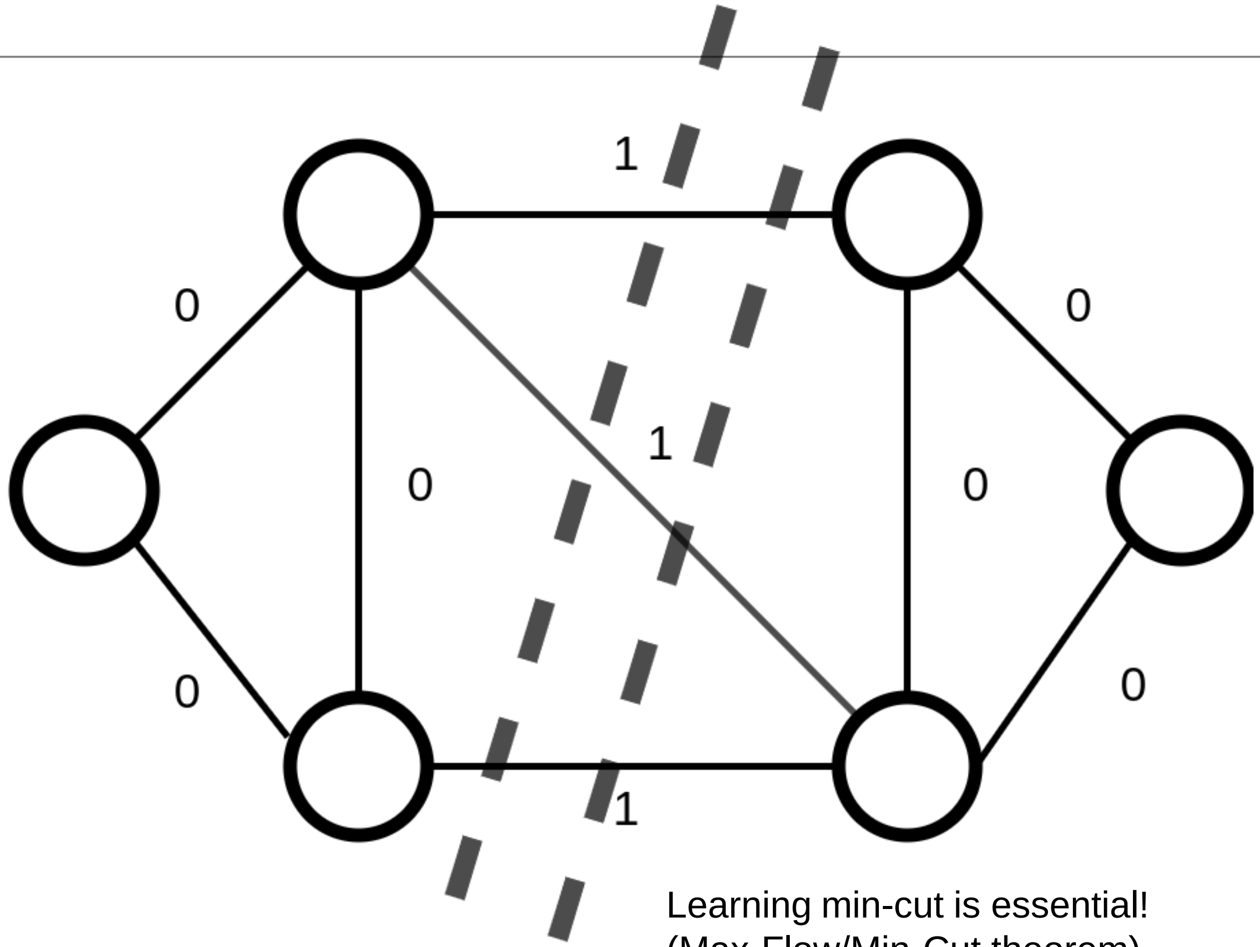
Numeroso, Bacciu, Velickovic, ICLR 2023

Università di Pisa

Example: Ford-Fulkerson, Max-Flow & Min-Cut



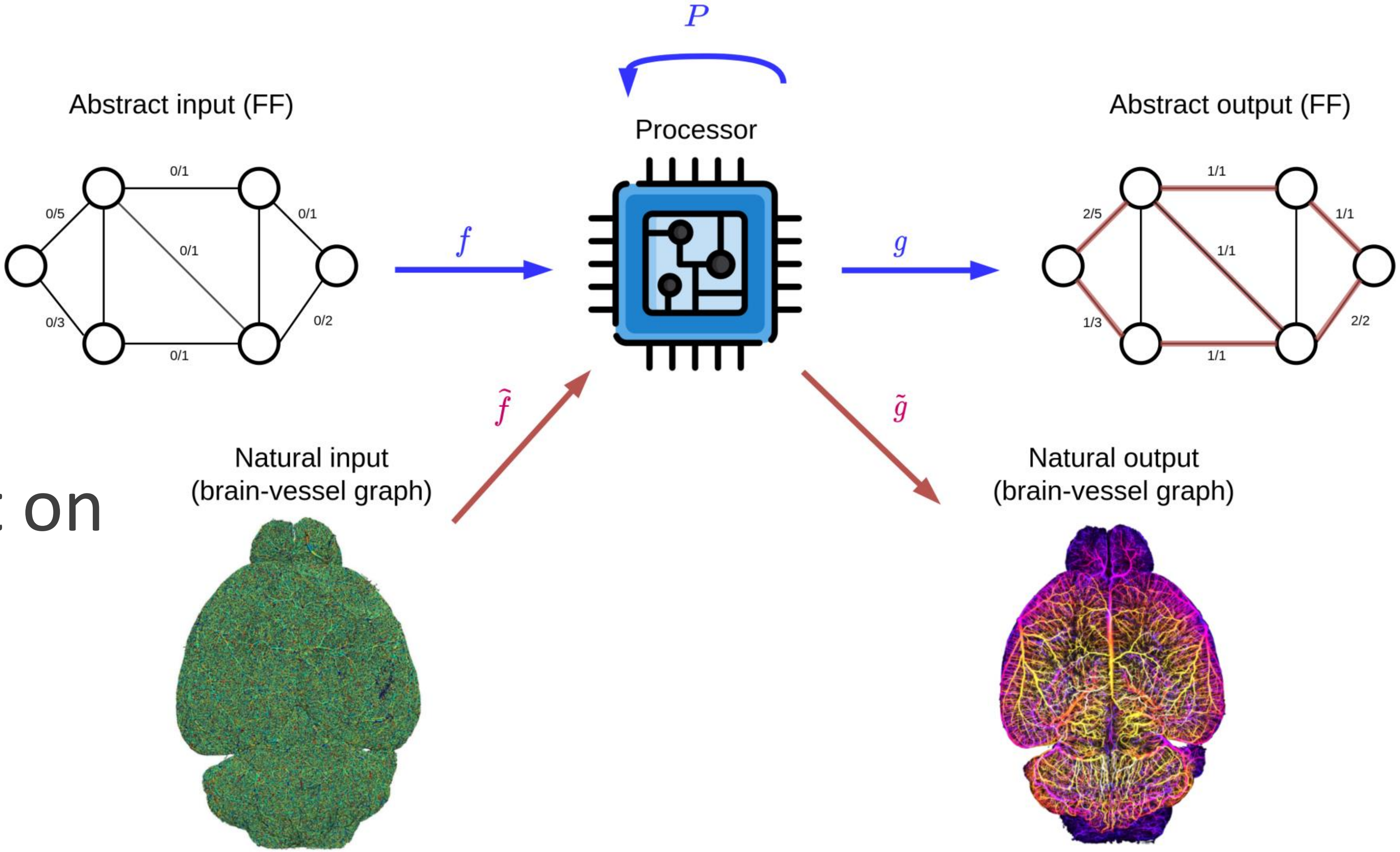
Example: Ford-Fulkerson, Max-Flow & Min-Cut



Learning min-cut is essential!
(Max-Flow/Min-Cut theorem)

Scaling up way out of distribution

Numeroso, Bacciu, Velickovic, ICLR 2023



Train on 16 nodes, predict on 10 millions

Wrap-Up

Software

You can find most of the foundational models in this lecture implemented here

PyTorch geometric

DeepGraphLibrary

PyDGN

Our Python library for Deep Graph Networks

github.com/diningphil/PyDGN

Data (Benchmarks)

OPEN GRAPH BENCHMARK

- Pytorch Geometric and DGL integration
- Standardized splits and evaluators + leader-board
- Node, link and graph property prediction tasks

TUDataset

- Standardise assessment of existing benchmarks rather than inventing new ones
- Chemical, social, vision, synthetic, bioinformatics (with leader-board)
- Pytorch Geometric and DGL integration

Conclusions

- Deep learning for graphs is a now a consolidated research area
  - DGNs as natural extensions of convolutional and recurrent architectures to graphs
  - A candidate AI model for the integration of symbolic knowledge, numerical data and reasoning
- First wave of works (now almost over?) focusing mainly on
  - Different ways of implementing message passing and aggregation on static graphs
  - Graph reductions and pooling
  - Expressivity properties associated with different aggregation functions
  - Efficiency and efficacy of context creation and propagation
- New wave of works focusing on
  - Dynamic graphs
  - DGNs as dynamical systems and their physical interpretation
  - Learning and aligning with (graph) algorithms
  - Oversmoothing, oversquashing and problems of the sort
- …in other words, plenty of opportunities for thesis work!

Next Lecture

Tomorrow 15/05 h.16.00

- Beyond accuracy: auditing LLMs based on exams designed for humans
- Guest Lecture by Wagner Meira