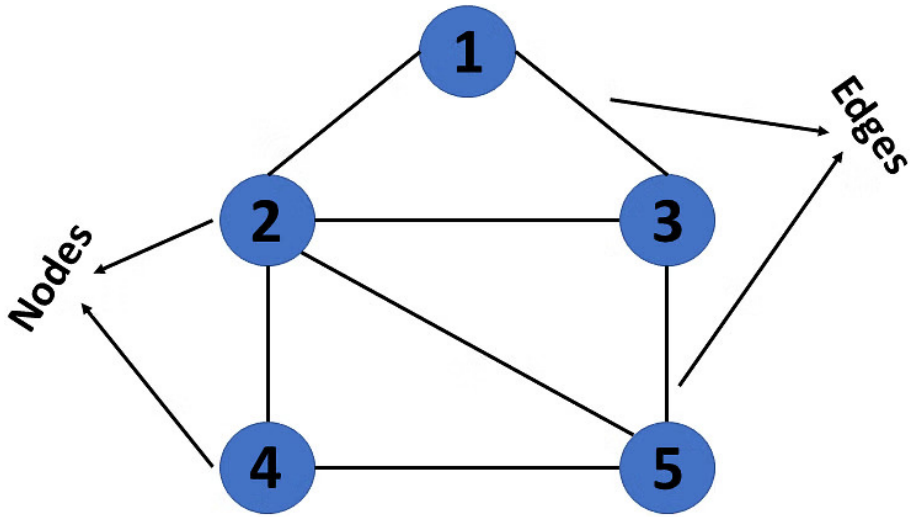
Graph Echo State Network

Jacopo Raffi

University of Pisa
MsC Computer Science - Curriculum AI
Computational Neuroscience A.Y. 2023/2024

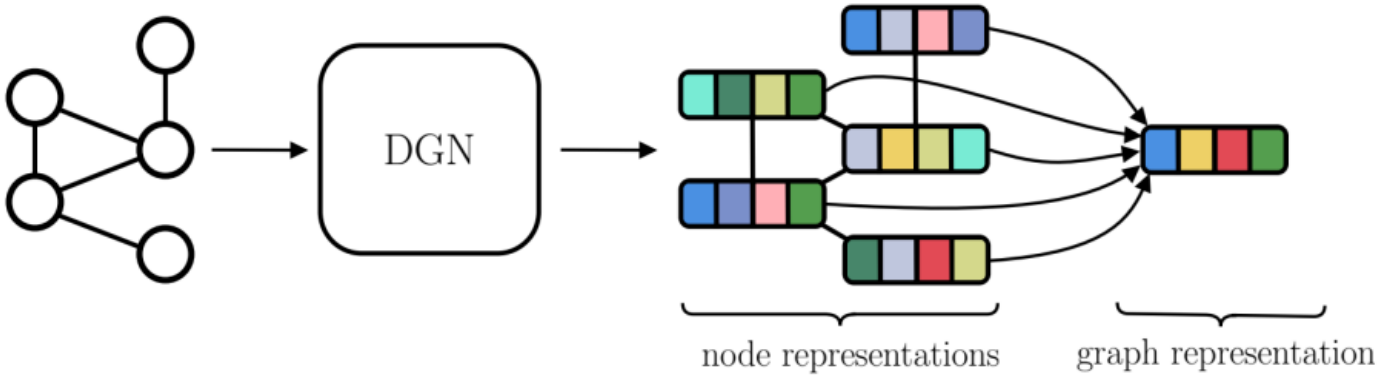
Introduction

Graph Structured Data



- Graphs are mathematical structures that represent relationships between objects;
- Examples of applications include biological systems, social networks, etc.

Deep Learning for Graphs



- Encode the nodes and the graph (if necessary) into a vector space;
- A node vector embedding needs to depend on its context;
- Different approaches based on recurrent [1] or feedforward architecture [2].

Graph Echo State Network

GraphESN

- Efficient recurrent-architecture based approach;
- Graph Echo State Network (**GraphESN**) [3] is a generalization of the Echo State Network approach to graph domain;
- The model consists of a reservoir layer (fixed) and a readout part (trainable).

Reservoir

Encoding Function

$$x_t(v) = f(W_{in}u(v) + W_H x_{t-1}(N_v))$$

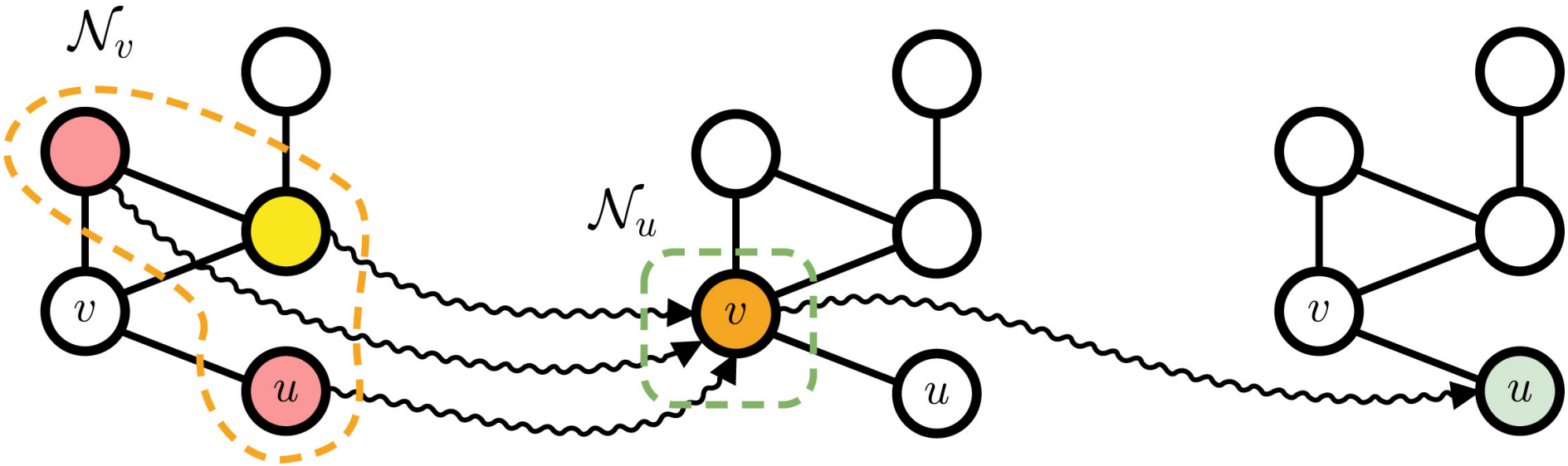
- $x_t(v)$ is the embedding of node v at time step t;
- $u(v)$ is the vector label of the node v, N_v is the neighbourhood of node v;
- W_{in} is the input weight matrix, W_H is the recurrent weight matrix;
- **Sufficient condition**, assuming f as $tanh$, for the contractivity of the encoding function is $\sigma < 1$, where $\sigma = (max_g|V(g)|)\ ||W_H||_2$, σ is called **contraction coefficient**;
- W_{in} and W_H are randomly initialized and then W_H is re-scaled to satify $\sigma < 1$.

Reservoir

Algorithm 1 Iterative computation of the Reservoir layer

$t = 0$
for $v \in V(g)$ **do**
    $x_0(v) = 0$
end for
while $||x_t(g) - x_{t-1}(g)||_2 > \epsilon$ **do**
    $t = t + 1$
    for $v \in V(g)$ **do**
        $x_t(v) = f(W_{in}u(v) + W_N x_{t-1}(N(v)))$
    end for
end while

Reservoir



Readout

- Once the embedding of the nodes is obtained, it is given as input to the readout part:
  - For node-level tasks, the readout is applied to each individual node as $y(v) = W_{out}x(v)$;
  - For graph-level tasks, the readout is applied to the graph as $y(g) = W_{out}X(g)$, where $X(g)$ is the embedding of the whole graph obtained using a state-mapping function, e.g. sum, average over nodes, concatenation of sum, average and max over nodes etc.
- Training of W_{out} is performed using linear regression.

GraphESN Results

	AB	AB+C	AB+C+PS
GraphESN	**(72 ± 4)%**	**(82 ± 7)%**	**(82 ± 7)%**
GNN	-	-	90%
TILDE	77%	82%	-
RDBC	83%	82%	-

Table: Test accuracy on MUTAG dataset.

	MR	FR	MM	FM
GraphESN	**(57 ± 4)%**	**(65 ± 3)%**	**(67 ± 5)%**	**(58 ± 4)%**
MG-Kernel	(63 ± 1)%	(70 ± 1)%	(69 ± 1)%	(65 ± 1)%
OA-Kernel	(63 ± 2)%	(65 ± 1)%	(68 ± 2)%	(65 ± 1)%
EM-Kernel	(61 ± 2)%	(69 ± 1)%	(67 ± 1)%	(65 ± 1)%

Table: Test accuracy on PTC dataset.

GraphESN Advancements

Fast and Deep Graph Neural Networks

- Fast and Deep Graph Neural Networks (FDGNN) [4] model exploits deep architecture, but avoiding the additional computational cost;

Encoding Function

$$x_t^i(v) = tanh(W_{in}^i x^{i-1}(v) + \sum_{v' \in N(v)} W_H^i x_{t-1}^i(v')), \forall v \, x^0(v) = u(v)$$

- Readout part: $y = W_{out} \; tanh(\sum_{v' \in V(g)} W_\varphi x^L(v))$, where W_φ is randomly initialized and re-scaled to have unitary $||.||_2$ norm and L is the last hidden layer;

- **Sufficient condition** [5] for GES(Graph Embedding Stability) is $||W_H||\alpha < 1$, where α is the input graph spectral norm;

- **Necessary condition** [5] is $\rho(W_H)\alpha^* < 1$, where α^* is the input graph spectral radius.

FDGNN Results

	MUTAG	PTC	NCI1
FDGNN	$(88.51 \pm 3.77)\%$	$(63.43 \pm 5.35)\%$	$(77.81 \pm 1.62)\%$
GNN	$(80.49 \pm 0.81)\%$	-	-
WL	$(84.11 \pm 1.91)\%$	$(57.97 \pm 2.49)\%$	$(84.46 \pm 0.45)\%$

Table: Average test accuracy (and std) on 10 fold.

FDGNN	GNN	WL
0.56 ± 0.33	202.28 ± 166.87	1.16 ± 0.03

Table: Training time (seconds) required on MUTAG (10 fold).

Ring Reservoir

- Graph Ring-Reservoir Network (GRN) [6] model imposes a ring-constrained pattern of connectivity between recurrent neurons to connect them into a cycle, the weights are set in a deterministic manner;
- Minimal-Graph Network (MGN) [6] model is an extension of the GRN model, where the input weights have the same value ω and the signs are chosen deterministically:
  - These minimal architectures (GRN and MGN) allow effective exploration of hyperparameters with comparable performance to more complex models.

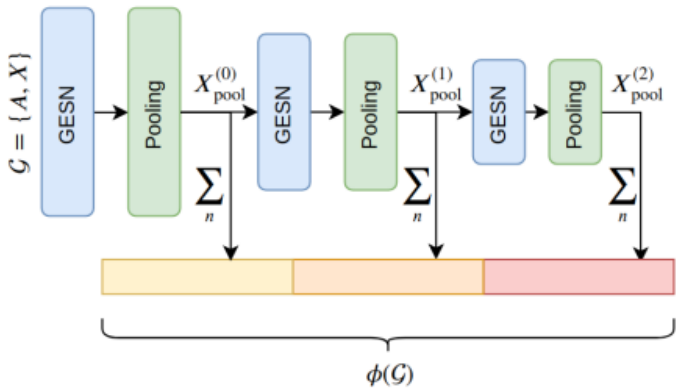
GRN and MGN Results

	MUTAG	NCI1	REDDIT	COLLAB
GESN(sparse)	88.2%	77.8%	87.5%	72.1%
FDGNN	88.5%	77.8%	89.5%	74.4%
GRN	88.4%	78.2%	87.6%	73.8%
MGN(reduced)	88.3%	77.6%	88.1%	74.1%
MGN(complete)	87.8%	78.8%	87.7%	74.5%

Table: Test accuracy on different datasets, MGN complete increase the number of explored configurations such that the total number of generated networks is the same as for GRN and GESN. MGN reduced same number of configurations as for GRN and GESN are explored.

Other GraphESN Advancements

- Stacking GraphESN layers with pooling layers [7], speed up the computation but slightly degrades performance;



- Multiresolution Reservoir Graph Neural Network (MRGNN) [8] model computes the reservoir iteration up to *k-hops* (avoid oversmoothing), concatenates all embeddings generated from the first to the last hop (multiresolution) to produce the final node embedding, which becomes the input to the MLP readout.

GraphESN Hardware

- Graph Neural Networks face signifcant challenges when running on conventional digital hardware (von Neumann bottleneck);
- The GraphESN hardware implementation [9] is based on resistive elements grouped into a crossbar array. These elements are capable of performing matrix multiplication, with the matrices themselves stored as the conductance of the resistive memory array. The multiplication and summation processes are enabled by Ohm's law and Kirchhoff's law. Notably, the data is stored and processed in the same location;
- State-of-the-art performance while reducing significantly the energy consumption e.g. training of GraphESN hardware version for MUTAG dataset consumes $34.51\mu J$ w.r.t the software version on a state-of-the-art GPU $74.32\mu J$ (53.56% reduction).

Conclusion

- GraphESN and its variants represent an efficient and well-founded ML approach to the graph domain;
- The results show a comparable, if not higher, performance various benchmarks;
- Possibility of hardware implementation to further reduce energy consumption;
- Extend the model to even larger graphs (distributed environment).

Thank you for your attention!

Bibliography I

[1] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini.
The graph neural network model.
IEEE Trans. Neural Networks, 20(1):61–80, 2009.

[2] Alessio Micheli.
Neural network for graphs: A contextual constructive approach.
IEEE Trans. Neural Networks, 20(3):498–511, 2009.

[3] Claudio Gallicchio and Alessio Micheli.
Graph echo state networks.
In *International Joint Conference on Neural Networks, IJCNN 2010, Barcelona, Spain, 18-23 July, 2010*, pages 1–8. IEEE, 2010.

Bibliography II

[4] Claudio Gallicchio and Alessio Micheli.
Fast and deep graph neural networks.
In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 3898–3905. AAAI Press, 2020.

[5] Domenico Tortorella, Claudio Gallicchio, and Alessio Micheli.
Spectral bounds for graph echo state network stability.
In *International Joint Conference on Neural Networks, IJCNN 2022, Padua, Italy, July 18-23, 2022*, pages 1–8. IEEE, 2022.

Bibliography III

[6] Claudio Gallicchio and Alessio Micheli.
Ring reservoir neural networks for graphs.
In *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*, pages 1–7. IEEE, 2020.

[7] Filippo Maria Bianchi, Claudio Gallicchio, and Alessio Micheli.
Pyramidal graph echo state networks.
In *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2020, Bruges, Belgium, October 2-4, 2020*, pages 573–578, 2020.

[8] Luca Pasa, Nicolò Navarin, and Alessandro Sperduti.
Multiresolution reservoir graph neural network.
IEEE Trans. Neural Networks Learn. Syst., 33(6):2642–2653, 2022.

Bibliography IV

[9] Shaocong Wang, Yi Li, Dingchen Wang, Woyu Zhang, Xi Chen, Danian Dong, Songqi Wang, Xumeng Zhang, Peng Lin, Claudio Gallicchio, Xiaoxin Xu, Qi Liu, Kwang-Ting Cheng, Zhongrui Wang, Dashan Shang, and Ming Liu.
Echo state graph neural networks with analogue random resistive memory arrays.
Nat. Mac. Intell., 5(2):104–113, 2023.

Appendix

GraphESN + Pooling results

	No-Pool		Graclus		NMF		NDP	
	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time
MUTAG	91.4	0.3s	88.9	0.5s	86.2	0.4s	89.4	0.2s
PROTEINS	77.9	5.3s	76.2	6.2s	73.6	2.7s	76.9	2.1s
NCI1	78.3	49s	72.6	31.7s	69.2	37.2s	74.9	35.4s

Table: Accuracy(%) and training time(seconds).

GRN and MGN structure

GRN and MGN recurrent weight matrix

$$\mathrm{P} = \begin{bmatrix} 0 & 0 & \dots & 1 \\ 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & 0 \end{bmatrix}$$

$W_H = \lambda P$, $\lambda = \rho/k$, k denotes the degree of the set of graphs under consideration.

MGN input weight matrix

$$W_{in} = \omega \prod$$

ω is a hyperparameter, $\prod$ is a sign matrix, whose entries are row-wise generated

$$\pi_i = \begin{cases} -1, & \text{if the corresponding digit of } \pi < 5 \\ +1, & \text{otherwise} \end{cases}$$