POLITECNICO DI TORINO
Repository ISTITUZIONALE

Edge-Wise Graph-Instructed Neural Networks

Original
Edge-Wise Graph-Instructed Neural Networks / Della Santa, F., Mastropietro, A., Pieraccini, S., Vaccarino, F.. - In: JOURNAL OF COMPUTATIONAL SCIENCE. - ISSN 1877-7503. - 85:(2025), pp. 1-6. [10.1016/j.jocs.2024.102518]

Availability:
This version is available at: 11583/2996523 since: 2025-01-11T15:24:47Z

Publisher:
Elsevier

Published
DOI:10.1016/j.jocs.2024.102518

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

25 November 2026

Journal of Computational Science 85 (2025) 102518

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Edge-Wise Graph-Instructed Neural Networks

Francesco Della Santa [a,b,*], Antonio Mastropietro [c], Sandra Pieraccini [a,b], Francesco Vaccarino [a]

[a] Department of Mathematical Sciences, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Turin, Italy
[b] Gruppo Nazionale per il Calcolo Scientifico INdAM, Piazzale Aldo Moro 5, 00185, Rome, Italy
[c] Department of Computer Science, University of Pisa, Largo B. Pontecorvo 3, 56127, Pisa, Italy

ARTICLE INFO

Dataset link: https://github.com/Fra0013To/GINN/tree/ewginn_dev/experiments/ewginn_paper_2024

MSC:
05C21
65D15
68T07
90C35

Keywords:
Graph neural networks
Deep learning
Regression on graphs

ABSTRACT

The problem of multi-task regression over graph nodes has been recently approached through Graph-Instructed Neural Network (GINN), which is a promising architecture belonging to the subset of message-passing graph neural networks. In this work, we discuss the limitations of the Graph-Instructed (GI) layer, and we formalize a novel edge-wise GI (EWGI) layer. We discuss the advantages of the EWGI layer and we provide numerical evidence that EWGINNs perform better than GINNs over some graph-structured input data, like the ones inferred from the Barabási-Albert graph, and improve the training regularization on graphs with chaotic connectivity, like the ones inferred from the Erdos–Rényi graph.

1. Introduction

Graph Neural Networks (GNNs) are powerful tools for learning tasks on graph-structured data [1], such as node classification [2], link prediction, or graph classification. Their formulation traces back to the late 2000s [3–5]. In the last years, GNNs have received increasing attention from the research community for their application in biology [6], chemistry [7,8], finance [9], geoscience [10], computational social science [11], and particle physics [12], to name a few. Among the available models in the literature, we mention Graph ConvNet, GraphSage, and Graph Attention Networks as models for tasks such as graph, node, or edge classification, or for graph regression [13,14]. Yet, the community has neglected the applications concerning the Regression on Graph Nodes (RoGN) learning task. Indeed, to the best of the authors' knowledge, the most used benchmarks do not include datasets for this task [13,14]. Nonetheless, there is an increasing interest in RoGN, especially among researchers working on physics-based simulations where, for example, predictions on mesh or grid nodes are performed (see for example [15]).

RoGN can be stated as multi-task regression, where the input data are endowed with a graph structure. The benchmark models for multi-task regression are Fully Connected Neural Networks (FCNNs). Recently, a new type of layer for GNNs has been developed in [10], belonging to the class of message-passing GNNs [8]. From now on we will refer to these layers as Graph-Instructed (GI) layers; Graph-Instructed NNs (GINNs) are built by stacking GI layers. GINNs have demonstrated good performance on RoGN, showing better results than FCNNs, as illustrated in [10]. Although the GINN architecture has been specifically designed for RoGN, the usage of GI layers has been recently extended to supervised classification tasks (see [16]).

We point the reader to the fact that in [10] GI layers and GINNs are denoted as Graph-Informed layers and Graph-Informed NNs, respectively. In [17], in a different framework from the one addressed in [10], a homonymous but different model is presented; therefore, to avoid confusion with [17], we have changed the names of both layers and NNs.

GI layers are based on a weight-sharing principle, such that their weights rescale the outgoing message from each node. In this paper, to improve the generalization capability of their inner-layer representation, we introduce Edge-Wise Graph-Instructed (EWGI) layers, characterized by additional weights (associated with graph nodes) that enable the edge-wise customization of the passage of information to each receiving node.

We compare the Edge-Wise GINN (EWGINN) with the GINN in the experimental settings originally used in [10] for validating the models;

* Corresponding author at: Department of Mathematical Sciences, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Turin, Italy.
E-mail address: francesco.dellasanta@polito.it (F. Della Santa).

https://doi.org/10.1016/j.jocs.2024.102518
Received 13 September 2024; Received in revised form 3 December 2024; Accepted 26 December 2024
Available online 2 January 2025
1877-7503/© 2025 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

these settings are RoGN tasks on two stochastic flow networks based on a Barabási-Albert graph and an Erdos–Rényi graph, respectively. In particular, we show that EWGINNs perform better on the Barabási-Albert connectivity structure, with a small increment of the number of learning weights.

The work is organized as follows: in Section 2 the GI layers are introduced, recalling their inner mechanisms. Section 3 formally defines EWGI layers and theoretically discusses their properties. Then, in Section 4, we analyze the experiment results for the RoGN tasks, comparing with the previous literature [10]. Finally, Section 5 summarizes our work and discusses future improvements and research directions.

2. Graph-instructed layers

This section briefly reviews previous GINNs to establish the framework for introducing our main contribution. Graph-Instructed (GI) Layers are NN layers defined by an alternative graph-convolution operation introduced in [10]. Given a graph G (without self-loops) and its adjacency matrix $A \in \mathbb{R}^{n\times n}$, a basic GI layer for G is a NN layer with one input feature per node and one output feature per node described by a function $\mathcal{L}^{GI} : \mathbb{R}^n \to \mathbb{R}^n$ such that

$$\mathcal{L}^{GI}(\boldsymbol{x}) = \boldsymbol{\sigma}\left((\mathrm{diag}(\boldsymbol{w})(A+\mathbb{I}_n))^T \boldsymbol{x} + \boldsymbol{b}\right), \tag{1}$$

for each vector of input features $\boldsymbol{x} \in \mathbb{R}^n$ and where:

- $\boldsymbol{w} \in \mathbb{R}^n$ is the weight vector, with the component w_i associated to the graph node v_i, $i = 1, \dots, n$.
- $\mathrm{diag}(\boldsymbol{w}) \in \mathbb{R}^{n\times n}$ is the diagonal matrix with elements of $\boldsymbol{w}$ on the diagonal and $\mathbb{I}_n \in \mathbb{R}^{n\times n}$ is the identity matrix. For future reference, we set $\widehat{W} := \mathrm{diag}(\boldsymbol{w})(A+\mathbb{I}_n)$;
- $\boldsymbol{\sigma} : \mathbb{R}^n \to \mathbb{R}^n$ is the element-wise application of the activation function σ;
- $\boldsymbol{b} \in \mathbb{R}^n$ is the bias vector.

In brief, Eq. (1) is equivalent to the action of a Fully-Connected (FC) layer where the weights are the same if the connection is outgoing from the same unit, whereas it is zero if two units correspond to graph nodes that are not connected; more precisely:

$$\widehat{w}_{ij} = \begin{cases} w_i, & \text{if } a_{ij} \neq 0 \text{ or } i = j \\ 0, & \text{otherwise} \end{cases},$$

where $a_{ij}, \widehat{w}_{ij}$ denote the (i,j)-th element of $A, \widehat{W}$, respectively.

On the other hand, from a message-passing point of view, the operation described in (1) is equivalent to having each node v_i of G sending to its neighbors a message equal to the input feature x_i, scaled by the weight w_i; then, each node sum up all the messages received from the neighbors, add the bias, and applies the activation function. In a nutshell, the message-passing interpretation can be summarized by the following node-wise equation:

$$x'_i = \sum_{j\in \mathrm{N}_{\mathrm{in}}(i)\cup\{i\}} x_j\, w_j + b_i, \tag{2}$$

where x'_i is the output feature of the GI layer corresponding to node v_i and $\mathrm{N}_{\mathrm{in}}(i)$ is the set of indices such that $j \in \mathrm{N}_{\mathrm{in}}(i)$ if and only if $e_{ij} = \{v_i, v_j\}$ is an edge of the graph. We dropped the action of the activation function σ for simplicity.

Layers characterized by (1) can be generalized to read any arbitrary number $K \geq 1$ of input features per node and to return any arbitrary number $F \geq 1$ of output features per node. Then, the general definition of a GI layer is as follows.

Definition 2.1 (*GI Layer — General form [10]*)**.** A *GI layer with* $K \in \mathbb{N}$ *input features and* $F \in \mathbb{N}$ *output features* is a NN layer with nF units connected to a layer with outputs in $\mathbb{R}^{n\times K}$ and having a characterizing function $\mathcal{L}^{GI} : \mathbb{R}^{n\times K} \to \mathbb{R}^{n\times F}$ defined by

$$\mathcal{L}^{GI}(X) = \boldsymbol{\sigma}\left(\widetilde{\mathbf{W}}^T \mathrm{vertcat}(X) + B\right), \tag{3}$$

where:

- $X \in \mathbb{R}^{n\times K}$ is the input matrix (i.e., the output of the previous layer) and $\mathrm{vertcat}(X)$ denotes the vector in $\mathbb{R}^{nK}$ obtained concatenating the columns of X;
- tensor $\widetilde{\mathbf{W}} \in \mathbb{R}^{nK\times F\times n}$ is the concatenation along the 2nd dimension (i.e., the column-dimension) of the matrices $\widetilde{W}^{(1)}, \dots, \widetilde{W}^{(F)}$, defined as

$$\widetilde{W}^{(l)} := \begin{bmatrix} \widehat{W}^{(1,l)} \\ \vdots \\ \widehat{W}^{(K,l)} \end{bmatrix} = \begin{bmatrix} \mathrm{diag}(\boldsymbol{w}^{(1,l)})\widehat{A} \\ \vdots \\ \mathrm{diag}(\boldsymbol{w}^{(K,l)})\widehat{A} \end{bmatrix} \in \mathbb{R}^{nK\times n}, \tag{4}$$

  for each $l = 1, \dots, F$, after being reshaped as tensors in $\mathbb{R}^{nK\times 1\times n}$. Vector $\boldsymbol{w}^{(k,l)} \in \mathbb{R}^n$ is the weight vector characterizing the contribution of the k-th input feature to the computation of the l-th output feature of the nodes, for each $k = 1, \dots, K$, and $l = 1, \dots, F$; matrix $\widehat{A}$ denotes $A + \mathbb{I}_n$.
- the operation $\widetilde{\mathbf{W}}^T \mathrm{vertcat}(X)$ is a tensor-vector product and $B \in \mathbb{R}^{n\times F}$ is the matrix of the biases.

Additionally, pooling and mask operations can be added to GI layers (see [10] for more details).

From now on, we call *Graph-Instructed Neural Network* (GINN) a NN made of GI layers [10]. We point out that the number of weights of a GI layer is equal to $nKF + nF$. On the other hand, the number of weights of a FC layer of n units, reading the outputs of a layer of m units, is equal to $mn + n$; therefore, if we consider the case of $m = n$ and $KF + F < n + 1$ (typically satisfied for sufficiently large graphs), GI layers have fewer weights to be trained compared with the FC layer. Moreover, we observe that adjacency matrices are typically sparse and, therefore, the tensor $\widetilde{\mathbf{W}}$ in (7) is typically sparse too. Then, it is possible to exploit the sparsity of this tensor to reduce the memory cost of the GINN implementation.

3. Edge-wise graph-instructed layers

A possible drawback of GI layers is that their weights rescale only the outgoing information of the nodes. For example, if nodes v_j and v_k are connected to node v_i in a graph $G = (V, E)$ such that $(v_i, v_j), (v_i, v_k) \in E$, then the units corresponding to v_j and v_k in a GI layer based on G receive the same contribution from the input features corresponding to node v_i; moreover, if nodes v_j, v_k have the same neighbors, the GI layer's outputs corresponding to these nodes are the same except for the contribution of the bias and the contribution from themselves. This property is useful to reduce the number of weights per layer and, depending on the complexity of the target function defined on the graph nodes, it is not necessarily a limitation. Nonetheless, it surely limits the representational capacity of the model. Therefore, some target functions can be too complicated to be modeled by GI layers.

Given the observation above, it is useful to define a new GI layer capable of improving the capacity of the model at a reduced cost in terms of the total number of trainable weights. In this work, we propose to modify the classic GI layers by adding an extra set of weights associated with the nodes to rescale their incoming information. In brief, given the node-wise Eq. (1), we change it into

$$x'_i = w_i^{\mathrm{in}} \sum_{j\in \mathrm{N}_{\mathrm{in}}(i)\cup\{i\}} x_j\, w_j^{\mathrm{out}} + b_i, \tag{5}$$

where w_j^{out} denotes the (old) weights for rescaling the outgoing information from node v_j, while w_i^{in} denotes the (new) weights for rescaling the incoming information to node v_i (see Fig. 1).

A NN layer based on (5) is a layer with one input feature per node and one output feature per node, described by a function $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}^n$ such that

$$\mathcal{L}(\boldsymbol{x}) = \boldsymbol{\sigma}\left((\mathrm{diag}(\boldsymbol{w}^{\mathrm{out}})(A+\mathbb{I}_n)\,\mathrm{diag}(\boldsymbol{w}^{\mathrm{in}}))^T \boldsymbol{x} + \boldsymbol{b}\right) \tag{6}$$

for each vector of input features $\boldsymbol{x} \in \mathbb{R}^n$ and where $\boldsymbol{w}^{\mathrm{out}}, \boldsymbol{w}^{\mathrm{in}} \in \mathbb{R}^n$ are the weight vectors, where the components $w_i^{\mathrm{out}}, w_i^{\mathrm{in}}$ are associated to

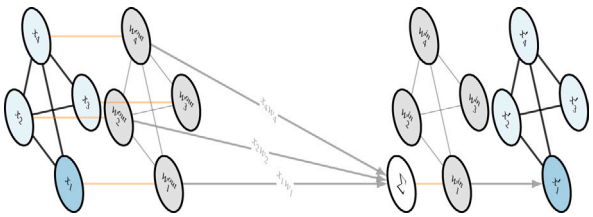


Fig. 1. Visual representation of (5). Example with $n = 4$ nodes (non-directed graph), $i = 1$; for simplicity, the bias is not illustrated.

the graph nodes v_i, for each $i = 1, \dots, n$. For future reference, from now on, we set $\widehat{W} := \mathrm{diag}(\boldsymbol{w}^{\mathrm{out}})(A + \mathbb{I}_n)\,\mathrm{diag}(\boldsymbol{w}^{\mathrm{in}})$.

In brief, (6) is equivalent to a FC layer where the weights are zero if two distinct units correspond to graph nodes that are not connected, otherwise $\widehat{w}_{ij} = w_i^{\mathrm{out}} w_j^{\mathrm{in}}$ if $e_{ij} \in E$ or $i = j$. Therefore, we observe that each weight $\widehat{w}_{ij} = w_i^{\mathrm{out}} w_j^{\mathrm{in}}$ is associated with the edge $e_{ij} = (v_i, v_j)$ in the graph or the self-loop added by the layer (if $i = j$). Given the above observations, we can interpret (6) as the operation of a NN layer with weights associated with edges instead of nodes. Then, we define the new layer as *Edge-Wise GI* (EWGI) Layer.

Remark 3.1 (*EWGI Layers — Advantages of the Formulation*)**.** Note that in principle EWGI layers could be defined by associating an independent weight $\widehat{w}_{ij}$ to each edge of G and each added self-loops. Nonetheless, the approach here proposed exhibits the following advantages:

- If G is a directed graph, we have that $n - 1 < |E| < n^2 - n$; therefore, for the independent weight formulation the total number of weights is in the range $[2n - 1, n^2]$ (biases excluded). On the other hand, in (6) the number of weights is always equal to $2n$ (biases excluded).
- If G is an undirected graph, we have $n - 1 < |E| < (n^2 - n)/2$; therefore, for the independent weight formulation the total number of weights is in the range $[2n - 1, n + (n^2 - n)/2]$ (biases excluded). On the other hand, in (6) the number of weights is always equal to $2n$ (biases excluded).

The advantage of using formulation (6) is therefore evident: independently of the number of graph edges, the number of weights is always $2n$, which is essentially the lower bound of the number of weights in the other formulation.

Analogously to classic GI layers, EWGI layers can be generalized to read any arbitrary number $K \geq 1$ of input features per node and to return any arbitrary number $F \geq 1$ of output features per node. Then, the general definition of a EWGI layer is as follows.

Definition 3.1 (*EWGI Layer — General Form*)**.** An *EWGI layer with* $K \in \mathbb{N}$ *input features and* $F \in \mathbb{N}$ *output features* is a NN layer with nF units connected to a layer with outputs in $\mathbb{R}^{n\times K}$ and having a characterizing function $\mathcal{L}^{EWGI} : \mathbb{R}^{n\times K} \to \mathbb{R}^{n\times F}$ defined by

$$\mathcal{L}^{EWGI}(X) = \boldsymbol{\sigma}\left(\widetilde{\mathbf{W}}^T \mathrm{vertcat}(X) + B\right), \tag{7}$$

where the tensor $\widetilde{\mathbf{W}} \in \mathbb{R}^{nK\times F\times n}$ is defined as the concatenation along the 2nd dimension of the matrices $\widetilde{W}^{(1)}, \dots, \widetilde{W}^{(F)}$, such that

$$\widetilde{W}^{(l)} := \begin{bmatrix} \mathrm{diag}(\boldsymbol{w}_{\mathrm{out}}^{(1,l)})\,\widehat{A}\,\mathrm{diag}(\boldsymbol{w}_{\mathrm{in}}^{(1,l)}) \\ \vdots \\ \mathrm{diag}(\boldsymbol{w}_{\mathrm{out}}^{(K,l)})\,\widehat{A}\,\mathrm{diag}(\boldsymbol{w}_{\mathrm{in}}^{(K,l)}) \end{bmatrix} \in \mathbb{R}^{nK\times n}, \tag{8}$$

for each $l = 1, \dots, F$, after being reshaped as tensors in $\mathbb{R}^{nK\times 1\times n}$, and where:

- $\boldsymbol{w}_{\mathrm{out}}^{(k,l)} \in \mathbb{R}^n$ is the weight vector characterizing the contribution of the k-th input feature to the computation of the l-th output feature of the nodes, for each $k = 1, \dots, K$, and $l = 1, \dots, F$, with respect to the outgoing information;
- $\boldsymbol{w}_{\mathrm{in}}^{(k,l)} \in \mathbb{R}^n$ is the weight vector characterizing the contribute of the k-th input feature to the computation of the l-th output feature of the nodes, for each $k = 1, \dots, K$, and $l = 1, \dots, F$, with respect to the incoming message.

From the definition above, we observe that the number of weights of a general EWGI layer is $2nKF + nF$. Therefore, if we consider a FC layer of n units, reading the outputs of a layer of $m = n$ units, the EWGI layers have a smaller number of weights to be trained if $2KF + F < n + 1$.

From now on, we call *Edge-Wise Graph-Instructed Neural Network* (EWGINN) a NN made of EWGI layers.

4. Preliminary results

In this section, we illustrate the results of a preliminary experimental study about the representational capacity of the new EWGI layers and EWGINNs. We compare the performances of a set of EWGINNs with the ones of a set of GINNs for the RoGN task of the two stochastic maximum flow problems reported in [10]. In particular, we train the models using the same architectures, hyperparameters, and training options; for the EWGINNs, we replace GI layers with EWGI layers. In order to strengthen the study, we train each configuration with respect to five different random seeds, reporting the median performances for each configuration.

4.1. Maximum flow regression for stochastic flow networks

Concerning the regression problem, we recall that a stochastic maximum-flow problem is a problem where the edge capacities in a flow network are modeled as random variables and the target is to find the distribution of the maximum flow (e.g., see [18]). The task is to approximate with a NN model the function

$$\begin{aligned} &\boldsymbol{\Phi} : \mathbb{R}_+^n \longrightarrow \mathbb{R}_+^m \\ &\quad \boldsymbol{c} \longmapsto \boldsymbol{\Phi}(\boldsymbol{c}) = \boldsymbol{\varphi} \end{aligned} \tag{9}$$

where $\boldsymbol{c} := (c_1, \dots, c_n) \in \mathbb{R}_+^n$ is the vector of the *capacities* of all the n edges of the network and $\boldsymbol{\varphi} := (\varphi_1, \dots, \varphi_m) \in \mathbb{R}_+^m$ is the flow vector corresponding to the m incoming edges of the network's sink that generate the maximum flow; in other words, the maximum flow corresponding to $\boldsymbol{c}$ is $\varphi := \|\boldsymbol{\Phi}(\boldsymbol{c})\|_1 = \sum_{j=i}^m \varphi_j$.

To address this regression task, we build the GINNs and the EWGINNs with respect to the adjacency matrix of the *line graph* of the flow network; i.e., on the graph where the vertices correspond to edges of the network and two vertices are connected if the corresponding edges in the network share at least one vertex. We refer to [10] for more details about the formulation of this RoGN task for learning the maximum flow of a stochastic flow network (SFN).

4.2. Performance measures

Let $\widehat{\boldsymbol{\Phi}}$ denote a NN model trained for learning (9) and let $\mathcal{P}$ be a test set used for measuring the performances of the model. Then, denoted by $\widehat{\boldsymbol{\varphi}} := \widehat{\boldsymbol{\Phi}}(\boldsymbol{c}) \in \mathbb{R}_+^m$, we define the following performance measures:

- Average Mean Relative Error (MRE) of sink's incoming flows, with respect to the max-flow:
$$\mathrm{MRE}_{av}(\mathcal{P}) := \frac{1}{m}\sum_{j=1}^m \left(\frac{1}{|\mathcal{P}|}\sum_{(\boldsymbol{c},\boldsymbol{\varphi})\in\mathcal{P}} \frac{|\varphi_j - \widehat{\varphi}_j|}{\varphi}\right). \tag{10}$$
This error measure describes the average quality of the NN in predicting the single elements $\varphi_1, \dots, \varphi_m$.
- Average max-flow MRE:
$$\mathrm{MRE}_{\varphi}(\mathcal{P}) : \frac{1}{|\mathcal{P}|}\sum_{(\boldsymbol{c},\boldsymbol{\varphi})\in\mathcal{P}} \frac{|\varphi - \widehat{\varphi}|}{\varphi} \tag{11}$$
This error measure describes the NN capability to predict the vector of fluxes $\widehat{\boldsymbol{\varphi}}$ such that the corresponding maxflow $\widehat{\varphi}$ approximates the true maxflow φ.

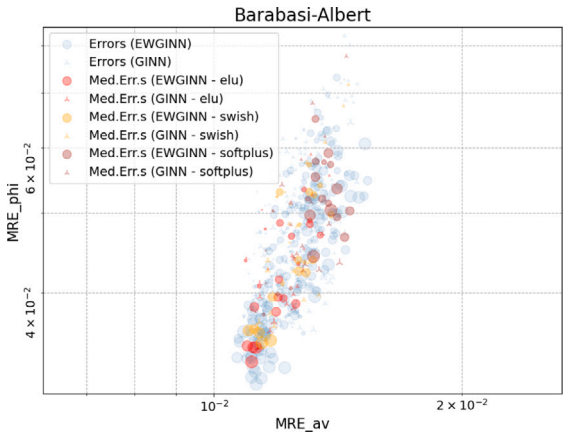


Fig. 2. $\mathcal{G}_{\mathrm{BA}}$. Performances of GINN and EWGINN models in the $(\mathrm{MRE}_{av}, \mathrm{MRE}_{\varphi})$ plane. Marker sizes are proportional to the number of NN weights.

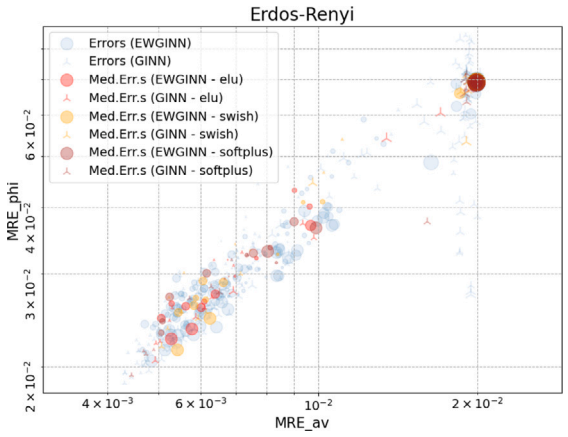


Fig. 3. $\mathcal{G}_{\mathrm{ER}}$. Performances of GINN and EWGINN models in the $(\mathrm{MRE}_{av}, \mathrm{MRE}_{\varphi})$ plane. Marker sizes are proportional to the number of NN weights.

4.3. Data, model architectures, and hyperparameters

We run our experiments on the same data reported in [10] for two randomly generated SFNs: a network based on a Barabási-Albert (BA) graph and a network based on an Erdos–Rényi (ER) graph. Each of the datasets D_{BA} and D_{ER} consists of 10 000 samples of capacity vectors and corresponding flow vectors.

In this work, we focus on the harder case illustrated in [10]: for each SFN, we train the EWGINN and GINN models on 500 samples (20% used as validation set), measuring the errors MRE_{av} and MRE_{φ} on a test set of 3000 samples. Then, we compare the performances obtained by EWGINNs and GINNs, looking at the "median models" of each training configuration, where the median is computed with respect to the five initializations generated through the different random seeds.

For a fair comparison, the architectures and hyperparameters of the EWGINN and GINN models are the same and follow the criteria indicated in [10]. Specifically, we build 60 EWGINN and GINN models configurations, respectively, for each SFN, varying among these parameters: hidden layers' activation function $\sigma \in \{ELU, swish, softplus\}$, depth $H \in \{3, 5, 7, 9\}$ for $\mathcal{G}_{\mathrm{BA}}$ and $H \in \{4, 9, 14, 19\}$ for $\mathcal{G}_{\mathrm{ER}}$, output features of each EWGI/GI layer $F \in \{1, 5, 10\}$, output layer's pooling operation (if $F > 1$) $pool \in \{reduce_max, reduce_mean\}$. Also, the training options are the same used in [10]: Adam optimizer [19] (learning rate ϵ=0.002, moment decay rates $\beta_1 = 0.9, \beta_2 = 0.999$), early stopping regularization [20] (550 epochs of patience, starting epoch 200, restore best weights), reduction on plateau for the learning rate (reduction factor $\alpha = 0.5$, 50 epochs of patience, minimum $\epsilon = 10^{-6}$). Each model configuration is trained five times, with respect to five different random seeds, respectively, for a total number of 1200 trained models (600 per SFN).

4.4. Analysis of the results

Figs. 2 and 3 compare the errors between classical GINNs (tripod markers) and EWGINN (circular markers). The error plane shows the MRE_{av} error on the x-axis and the MRE_{φ} on the y-axis. The dot sizes are proportional to the number of NN weights, and dots corresponding to "median models" are colored according to the activation functions. Each median model is computed, among the five random seeds for each configuration, with respect to the distance of $(\mathrm{MRE}_{av}, \mathrm{MRE}_{\varphi})$ from the origin of the plane.

We observe that the performance of GINNs and EWGINNs are comparable both in $\mathcal{G}_{\mathrm{BA}}$ and $\mathcal{G}_{\mathrm{ER}}$, but different behaviors characterize them. In $\mathcal{G}_{\mathrm{BA}}$ (Fig. 2), the comparison is almost straightforward: GINNS and EWGINNs show similar trends in their performance, varying configurations and random seeds; however, EWGINNs show general better performance than GINNs for this SFN.

On the other hand, in $\mathcal{G}_{\mathrm{ER}}$ (Fig. 3), the performance trends of GINNs and EWGINNs are different. In particular, we observe that the EWGINN performances appear "more stable" than the GINN ones, varying configurations and random seeds. Indeed, we observe that EWGINN error points present a rather compact distribution, showing good regularization abilities of EWGINNs on the RoGN task (i.e., they reduce equally both the errors); on the contrary, GINN errors exhibit a sparser distribution; then, these models sometimes learn the task focusing more on MRE_{φ} than MRE_{av}. We point out that the ability to learn the RoGN task without preferences in reducing one of the two errors is well appreciated. Indeed, as observed in [10], a small MRE_{φ} and large MRE_{av} can be the result of symmetric underestimation/overestimation of the single flows. Therefore, even if the best performances are reached by a subset of GINNs, the EWGINNs prove to be more reliable, varying hyperparameters and initializations, while maintaining very good performances. These observations in $\mathcal{G}_{\mathrm{ER}}$ can be explained by the more chaotic connection structure of the SFN, if compared to $\mathcal{G}_{\mathrm{BA}}$; Indeed, EWGINNs has a clear advantage in regularizing their training, thanks to the property of rescaling the incoming information of nodes through additional weights.

We conclude by observing a cluster of GINNs and EWGINNs with poor performances for $\mathcal{G}_{\mathrm{ER}}$, constrained in an extremely small region (top-right corner, Fig. 3); the reason is an issue with early stopping. Specifically, a relatively fast reduction of the validation loss (VL) happens, resulting in a temporary overfitting or non-decreasing-VL phenomenon. This induces an interruption of the training due to the early stopping. Nonetheless, by removing the early stopping and increasing the training epochs, we observe that the overfitting phenomenon tends to disappear (see Fig. 4); moreover, in some cases, the VL starts to decrease again after some epochs. Therefore, we conclude that the larger representational capacity of EWGINNs is an advantage but requires more careful tuning of the training hyperparameters. On the contrary, the GINNs are less influenced by this behavior because of their reduced size; nonetheless, when they "escape" from such situations, they usually fall into the problem of focusing more on MRE_{φ} than MRE_{av}.

We defer to future work an in-depth analysis of EWGINNs by varying the training hyperparameters, such as the early stopping patience.

5. Conclusion

In this work, we proposed a novel type of GI layer: the Edge-Wise GI layer. Compared with the original GI layers, each node of

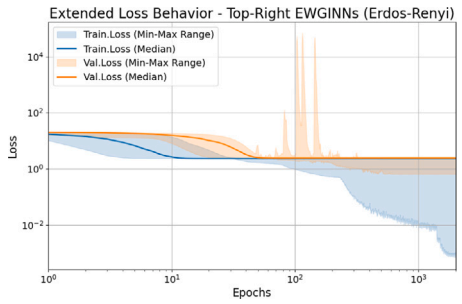


Fig. 4. Training and validation loss of the EWGINN corresponding to the top-rightmost dots in Fig. 3.

an EWGI layer is equipped with an additional weight for rescaling the incoming message. This enables improved representational capacity and breaks the symmetry of GI layers, where nodes with the same neighborhood invariably receive the same message from the previous layer. To analyze the performance of the newly proposed layers, we compared EWGINNs and GINNs on two benchmark RoGN tasks based on two SFNs, respectively: one with graph connectivity concentrated on a few more central nodes ($\mathcal{G}_{\text{BA}}$); one characterized by a random structure ($\mathcal{G}_{\text{ER}}$).

The numerical experiments show comparable performance between GINNs and EWGINNs on both SFNs, though we observe distinct behaviors. EWGINNs perform better than GINNs on $\mathcal{G}_{\text{BA}}$; on the other hand they exhibit improved regularization abilities on $\mathcal{G}_{\text{ER}}$, maintaining comparable performance with GINNs. These results highlight the advantages of EWGINNs, particularly in handling the chaotic structure of $\mathcal{G}_{\text{ER}}$, though their larger representational capacity demands more careful hyperparameter tuning. Observations of poor performance models caused by a too-early stopping suggest future studies focused on optimizing training configurations for EWGINNs. Future work will focus on applications to real-world problems.

CRediT authorship contribution statement

Francesco Della Santa: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Antonio Mastropietro:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Sandra Pieraccini:** Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Francesco Vaccarino:** Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Code availability

The code for implementing the EWGI layers introduced in this paper is available at: https://github.com/Fra0013To/GINN/tree/ewginn_dev.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

F.D., S.P., and F.V. acknowledge that this study was carried out within the FAIR-Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR)–MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3—D.D. 1555 11/10/2022, PE00000013). A.M. acknowledges support from the FINDHR project that received funding from the European Union's Horizon Europe research and innovation program under grant agreement No. 101070212. This manuscript reflects only the authors' views and opinions; neither the European Union nor the European Commission can be considered responsible for them. F.D. and S.P. acknowledge support from Italian MUR PRIN project 20227K44ME, Full and Reduced order modeling of coupled systems: focus on non-matching methods and automatic learning (FaReX).

Data availability

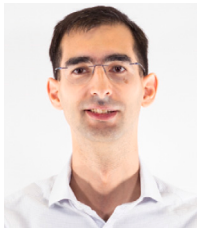
Link to the data is available at: https://github.com/Fra0013To/GINN/tree/ewginn_dev/experiments/ewginn_paper_2024.

References

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2021) 4–24, http://dx.doi.org/10.1109/TNNLS.2020.2978386.
[2] S.K. Maurya, X. Liu, T. Murata, Simplifying approach to node classification in graph neural networks, J. Comput. Sci. 62 (2022) 101695.
[3] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005, Vol. 2, 2005, pp. 729–734, http://dx.doi.org/10.1109/IJCNN.2005.1555942.
[4] A. Micheli, Neural network for graphs: A contextual constructive approach, IEEE Trans. Neural Netw. 20 (3) (2009) 498–511, http://dx.doi.org/10.1109/TNN.2008.2010350.
[5] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2009) 61–80, http://dx.doi.org/10.1109/TNN.2008.2005605.
[6] P. Cinaglia, Multilayer biological network alignment based on similarity computation via graph neural networks, J. Comput. Sci. 78 (2024) 102259.
[7] K. Atz, F. Grisoni, G. Schneider, Geometric deep learning on molecular representations, Nat. Mach. Intell. 3 (12) (2021) 1023–1032.
[8] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: International Conference on Machine Learning, PMLR, 2017, pp. 1263–1272.
[9] D. Cheng, F. Yang, S. Xiang, J. Liu, Financial time series forecasting with multi-modality graph neural network, Pattern Recognit. 121 (2022) 108218.
[10] S. Berrone, F. Della Santa, A. Mastropietro, S. Pieraccini, F. Vaccarino, Graph-informed neural networks for regressions on graph-structured data, Math. 10 (5) (2022) 786, http://dx.doi.org/10.3390/math10050786.
[11] S. Aref, M. Mostajabdaveh, Analyzing modularity maximization in approximation, heuristic, and graph neural network algorithms for community detection, J. Comput. Sci. 78 (2024) 102283.
[12] G. DeZoort, P.W. Battaglia, C. Biscarat, J.-R. Vlimant, Graph neural networks at the Large Hadron Collider, Nat. Rev. Phys. 5 (5) (2023) 281–303.
[13] V.P. Dwivedi, C.K. Joshi, A.T. Luu, T. Laurent, Y. Bengio, X. Bresson, Benchmarking graph neural networks, J. Mach. Learn. Res. 24 (43) (2023) 1–48.
[14] V.P. Dwivedi, L. Rampasek, M. Galkin, A. Parviz, G. Wolf, A.T. Luu, D. Beaini, Long range graph benchmark, Adv. Neural Inf. Process. Syst. 35 (2022) 22326–22340.
[15] F. Pichi, B. Moya, J.S. Hesthaven, A graph convolutional autoencoder approach to model order reduction for parametrized PDEs, J. Comput. Phys. 501 (2024) 112762, http://dx.doi.org/10.1016/j.jcp.2024.112762, URL https://www.sciencedirect.com/science/article/pii/S0021999124000111.
[16] F. Della Santa, S. Pieraccini, Graph-informed neural networks for sparse grid-based discontinuity detectors, 2024, arXiv:2401.13652.
[17] E.J. Hall, S. Taverniers, M.A. Katsoulakis, D.M. Tartakovsky, GINNs: Graph-informed neural networks for multiscale physics, J. Comput. Phys. 433 (2021) 110192, http://dx.doi.org/10.1016/j.jcp.2021.110192, URL https://www.sciencedirect.com/science/article/pii/S0021999121000875.
[18] S. Ding, The α-maximum flow model with uncertain capacities, Appl. Math. Model. 39 (7) (2015) 2056–2063, http://dx.doi.org/10.1016/j.apm.2014.10.021, URL https://www.sciencedirect.com/science/article/pii/S0307904X14004946.
[19] D.P. Kingma, J.L. Ba, Adam: A method for stochastic optimization, in: International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 2015, pp. 1–15, arXiv:1412.6980.
[20] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, www.deeplearningbook.org.



Della Santa, Francesco Research Associate at Politecnico di Torino. He got a Master Degree in Mathematics at University of Florence and a Ph.D. in Pure and Applied Mathematics at Politecnico di Torino. His main scientific interests concern Deep Learning, Surrogate Models, Uncertainty Quantification, and Numerical Optimization.



Mastropietro, Antonio Research Fellow at University of Pisa. Graduated in Mathematical Engineering at Politecnico di Torino, he got a Ph.D. in Pure and Applied Mathematics at Politecnico di Torino. His research focuses on eXplainable AI, Deep Reinforcement Learning, Deep Learning applied to computer vision, Cooperative Game Theory, and Graph Neural Networks.



Pieraccini, Sandra Full Professor in Numerical Analysis at Politecnico di Torino. She got a Master Degree in Mathematics and a Ph.D. in Applied Mathematics and Operation Research. Her main scientific interests concern Numerical Optimization and Uncertainty Quantification; recently these interests have been combined by investigations in the field of deep learning. She is author of more than 50 scientific publications on international journals and conference proceedings. She is the head of the working group on "Mathematics for Artificial Intelligence and Machine Learning" in the Unione Matematica Italiana (Italian Mathematical Union).



Vaccarino, Francesco Associate Professor in Geometry at Politecnico di Torino. Francesco Vaccarino is a mathematician and is currently working on several aspects of topological data analysis and computational topology. He gave talks at the International Congress of Mathematicians – ICM 2006 – Madrid, the Ams – Summer School in Algebraic Geometry 2005, Netsci 2014, 2015, 2016, JMM AMS MAA 2017. As a scientific journalist, he is regularly publishing on Tuttoscienze and wrote, with Gabriele Beccaria , the book: "Prima che accada".