Engineering Applications of Artificial Intelligence 117 (2023) 105567

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Power flow analysis via typed graph neural networks

Tania B. Lopez-Garcia, José A. Domínguez-Navarro [*]

University of Zaragoza, C. María de Luna, Zaragoza, 50018, Spain

ARTICLE INFO

Keywords:
Power flow
Graph neural networks
Unsupervised learning
Physics-informed neural networks

ABSTRACT

Power flow analyses are essential for the correct operation of power grids, however, electricity systems are becoming increasingly complex to analyze with the conventional numerical methods. The present work proposes a typed graph neural network based approach to solve the power flow problem. The neural networks are trained on benchmark power grid cases which are modified by varying the injections (load and generation), branch characteristics and topology. The solution to the power flow analysis is found when all voltage values are known. The proposed system infers the voltage magnitude and phase and is trained so that the obtained values minimize the violation of the physical laws that govern the system, this way the training is achieved in an unsupervised manner. The proposed solver has linear time complexity and is able to generalize to grids with considerably different conditions, including size, from the grids available during training. Though the training is unsupervised and does not suppose any ground truth data, the solutions obtained are found to have a close correlation with the conventional Newton–Raphson method. The results are additionally validated by finding the root mean square deviation from the Newton–Raphson method, and the faster, though less accurate, DC approximation method.

1. Introduction

For correct power system operation, power flow analyses must be executed frequently, as they are necessary for many procedures such as power systems planning, security assessment, management and optimization, Glover et al. (2012). Conventionally, the power flow analysis is carried out by determining and solving a set of non-linear algebraic equations with iterative numerical analysis methods; most known methods of this type have been tested at some point to solve the power flow problem, Stott (1974) and van Amerongen (1989). In recent years both the importance and complexity of software modeling of the electrical grid have increased due to higher electrical demand and the need to increase the sustainability of the conventional power grid, Smith et al. (2022). These issues arise from the ambitious, yet necessary, goals for emission reduction. Electrification of diverse energy sectors, such as heating and transport, has become a strategy for decarbonization, this phenomenon has consequently increased the demand and the dependence on electricity, Xie et al. (2021). In parallel, the use of variable renewable energy sources (VRES), such as solar and wind, is expanding. VRES are non-dispatchable and require power systems to become more flexible; they may cause loading in sections of the grid where it is usually not expected and cause instability, Babatunde et al. (2020). Many of the methods currently used for power system modeling were developed before widespread integration of VRES and the electrification of transport and heating, furthermore, most are computationally intensive. The increasing complexity of the electricity system may become too convoluted to describe in a timely and precise manner with conventional PF analysis approaches, thus requiring the investigation of new tools for power system modeling (Tovar-Facio et al., 2021).

With the emergence of artificial intelligence, many systems such as decision trees, neural networks and fuzzy logic methods have been applied to power system problems, Vankayala and Rao (1993) and Lopez-Garcia et al. (2020). Amongst these approaches, artificial neural networks (NNs) have shown promise to a certain extent, due to their ability to synthesize complex mappings accurately and rapidly, along with the possibility to continuously learn. However, most NNs used for power system modeling, such as in Hu et al. (2021) and Fikri et al. (2018), implement multi-layer perceptrons (MLPs). MLPs usually suffer from local minima and over-fitting issues, in addition to not scaling well to larger power grids; MLP based models cannot be used for grids of different size or configuration from the ones they are trained on.

Recently, graph neural networks (GNNs) have gained popularity for learning structured data and transferring learned information beyond training conditions (Battaglia et al., 2018). These methods convey strong relational inductive biases by establishing the GNN architecture directly on the structure of the analyzed system, which guides these approaches towards learning not only about the elements of the system, but also the relationships between them. By basing the proposed solver on GNNs, several advantages are introduced, such as scaling well to

* Corresponding author.
E-mail addresses: lopez.200916947@gmail.com (T.B. Lopez-Garcia), jadona@unizar.es (J.A. Domínguez-Navarro).

https://doi.org/10.1016/j.engappai.2022.105567
Received 8 June 2022; Received in revised form 11 October 2022; Accepted 25 October 2022
Available online 15 November 2022

larger power grids, and having the ability to generalize decently on grids of different size and configuration from the grids seen during training.

There are a few other works that apply GNNs (or NNs in general) for power grid analysis. The work in Owerko et al. (2020) uses GNNs but is based on imitation learning, and thus the GNN is burdensome to train and does not generalize well outside similar cases to those seen in training. A pioneering work by Donon et al. (2020) also applies GNNs, and training is carried out in an unsupervised manner, however the model is complicated and does not consider changes in grid topology during training.

The proposed model presents a simple GNN based solver to calculate the AC power flow in a way that allows to solve many different scenarios in parallel, considering the continuously changing balance between energy supply and demand, and does this in linear time as opposed to the exponential time needed for conventional methods that solve Jacobian matrices. The proposed solver is based on a generalization of GNNs called typed graph networks (TGNs) (Avelar et al., 2019), which allows different types of elements to be defined instead of the usual nodes and edges. The use of TGNs allows to obtain more faithful representations of the different elements present in electrical grids, such as the branches and the different types of buses, by considering each of them as different node types in the corresponding graph representation. Like other GNN based models, the presented solver can generalize to electrical grids of different sizes and parameters, yet it additionally considers changes in the grid topology during training, making it specially adequate for analyzing different scenarios of possible line outages, which is essential for security assessments. Additionally, the training is carried out in an unsupervised manner, applying physics-informed neural networks by incorporating information of the physical system in the loss function and aiming to minimize the violation of the physical laws that govern the system, thus eliminating the time consuming need of solving the training cases beforehand with other solvers to produce targets. The solver is modular in nature, allowing to connect different node types in any desired configuration. Additionally the time complexity of the proposed solver is linear, in contrast with common methods, such as the Newton–Raphson, which has exponential time complexity.

In summary, the proposed method consists of a TGN based system that abstracts the relationship between the different elements in the electrical grid to solve the steady state power flow problem for dynamical networks, i.e. considering different grid configurations, injections and branch characteristics. Thus, the presented work presents a valuable step towards developing a machine learning based system that is able to assist in analyzing flexible electrical grids of increasing complexity, while improving speed and reducing the computational burden of essential power flow analyses.

2. Preliminaries

In this section the basic components and power flow formulation are described, along with a general description of TGNs, which are the basis of the proposed model and solver. The description of the variables used to describe the electrical grid can be found in Table 1, and the definition of the main components used in the proposed solver are found in Table 2.

2.1. Power flow formulation

This work focuses on solving the power flow problem of power grids in a steady-state condition. Power grids are basically formed by buses, branches, loads and generators. The buses are the nodes to which the other elements are connected. Branches connect two buses and are modeled internally using the standard π transmission line model. The branch series impedance is given by the complex value: $z_k = r_k + ix_k$, where the resistance of the branch model constitutes the real part, and the equivalent reactance, the imaginary part. The shunt susceptance of the equivalent branch model is represented by b_k; this general model can include an ideal phase shifting transformer located at the sender end of the branch. The objective of the power flow analysis is to determine the voltage magnitude and phase at all buses for a given load, generation, and grid configuration state.

Table 1
Power grid nomenclature.

Variable	Units	Definition
n	–	Bus ID index
m	–	Generator bus ID index
k	–	Branch ID index
from_k	–	Sender bus ID of branch k
to_k	–	Receiver bus ID of branch k
V_n	kV	Voltage magnitude at bus n
θ_n	rad	Voltage phase at bus n
Pd_n	MW	Active power load at bus n
Qd_n	MVAr	Reactive power load at bus n
Pg_m	MW	Active power generation at generation bus m
Vg_m	kV	Voltage magnitude setpoint at generation bus m
r_k	Ω	Series resistance for branch k
x_k	Ω	Series reactance for branch k
b_k	S	Total line charging susceptance for branch k
τ_k	–	Transformer tap ratio
θ_k^{shift}	rad	Transformer phase shift

The power that flows through the grid depends on the power imbalances at the buses and the impedance of the branches, while the power balance at each bus is determined by the possible loads, generators and branches attached to it. Loads and generators constitute the external injections of the power grid, loads as a specified power demand, and generators as a specified power source. Given the grid topology, the specified values of the injections and line characteristics, the proposed power flow solver computes the resulting voltages in the buses and hence the current flow through the branches can be determined. The relationship between the branch characteristics, the voltages of the buses and the currents is given by:

$$\begin{bmatrix} i_{\text{from}_k} \\ i_{\text{to}_k} \end{bmatrix} = Y_{br,k} \begin{bmatrix} v_{\text{from}_k} \\ v_{\text{to}_k} \end{bmatrix} \tag{1}$$

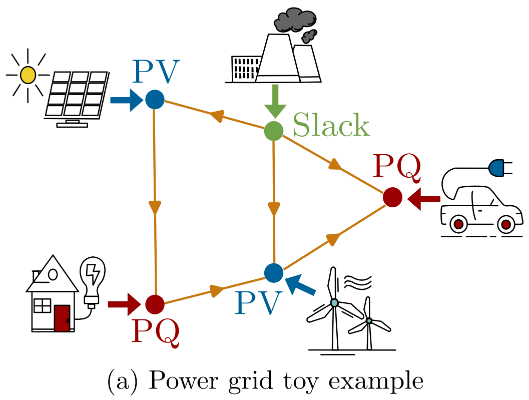
where i_{from_k} and i_{to_k} represent the complex current injections at the sender and receiver sides of branch k, respectively; v_{from_k} and v_{to_k} are the complex voltage values at the sender and receiver sides of branch k, respectively; the k branch admittance matrix $Y_{br,k}$ given by:

$$Y_{br,k} = \begin{bmatrix} \left(y_k + i\frac{b_k}{2}\right)\frac{1}{\tau_k^2} & -y_k \frac{1}{\tau_k e^{-i\theta_k^{\text{shift}}}} \\ -y_k \frac{1}{\tau_k e^{-i\theta_k^{\text{shift}}}} & y_k + i\frac{b_k}{2} \end{bmatrix} \tag{2}$$

where the series admittance element y_k is denoted by $y_k = \frac{1}{z_k}$.

It should be noted that before the load flow is solved, the network losses are unknown, thus, a generator bus, called the slack bus, is designated to compensate for these losses. The voltage magnitude and phase are given beforehand for the chosen slack bus ($n = 0$), and the power needed to compensate for the total grid losses must be determined. In addition to the slack bus, two other types of buses are defined: PV buses constitute the set of buses directly connected to a generator (that are not the slack bus); the remaining non-generation buses are classified as PQ buses. For each PV bus n_{PV}, the voltage magnitude is given by $Vg_{n_{PV}}$ and the generated active power is given by $Pg_{n_{PV}}$; for these buses the voltage phase $\theta_{n_{PV}}$, and the generated reactive power $Qg_{n_{PV}}$, must be determined. For each PQ bus n_{PQ}, the active and reactive powers are given by: $P_{n_{PQ}} = -Pd_{n_{PQ}}$ and $Q_{n_{PQ}} = -Qd_{n_{PQ}}$; for these buses both voltage magnitude and phase, $V_{n_{PQ}}$ and $\theta_{n_{PQ}}$, must be found.

For a solution to be obtained, the power balance in all nodes must be achieved by solving a non-linear equation system of the form $\Delta S = 0$,



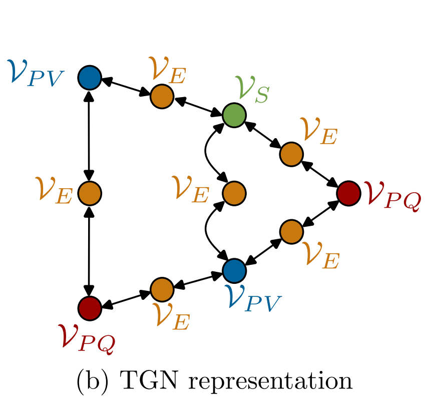


Fig. 1. Relationship between a power grid toy example and the corresponding TGN layer.

Table 2
Proposed solver components.

Component	Definition
$\mathcal{G}$	A graph
ζ	Set of node types
$\mathcal{V}_i$	Set of type i nodes
d_i	Embedding dimension of type i nodes
$\mathbf{A}$	Adjacency matrix
L	Total message passing iterations
T	Total TGN layers

which is deconstructed into nodal power balance equations as functions of unknown voltage values, as shown below for a bus n:

$$\Delta P_n(V_n,\theta_n) = \quad Pg_n - Pd_n - \mathrm{Re}\left[\left(\sum_{\substack{k\in\mathcal{N}(n)\\ n=\mathrm{from}_k}} i_{\mathrm{from}_k} + \sum_{\substack{k\in\mathcal{N}(n)\\ n=\mathrm{to}_k}} i_{\mathrm{to}_k}\right)\cdot v_n\right] \tag{3}$$

$$\Delta Q_n(V_n,\theta_n) = \quad Qg_n - Qd_n - \mathrm{Im}\left[\left(\sum_{\substack{k\in\mathcal{N}(n)\\ n=\mathrm{from}_k}} i_{\mathrm{from}_k} + \sum_{\substack{k\in\mathcal{N}(n)\\ n=\mathrm{to}_k}} i_{\mathrm{to}_k}\right)\cdot v_n\right] \tag{4}$$

where $k \in \mathcal{N}(n)$ represents all the branches that are connected to bus n.

There are $N_{pv} + 2N_{pq}$ voltage values that must be found (only phase for PV buses and both phase and magnitude for PQ buses), where N_{pv} and N_{pq} are the number of PV and PQ buses, respectively. Afterwards, $N_{pv} + 1$ reactive power balance equations are solved to find the generator reactive power injections, and this way all unknown variables are found.

2.2. Typed graph networks

It results straightforward to represent a power grid as a graph, the approach taken in this work is illustrated with a toy example in Fig. 1. The toy power grid shown in Fig. 1(a) includes PV, PQ and slack nodes, the branches that connect them and the external injections; Fig. 1(b) shows how the proposed solver architecture is directly related to the grid structure. The goal of the presented system is to infer the voltage values by representing the power grid as graph structured data and learning the relationships between the different elements; in other words, the proposed system takes a relational data graph as input, and outputs nodal voltage predictions. To achieve said goal, TGNs are employed; a brief description is found below.

Normally, a graph is defined as $\mathcal{G} = (\mathcal{V},\mathcal{E})$, with $\mathcal{V} = \{1,\dots,N\}$ a finite set of nodes, and $\mathcal{E} = \{1,\dots,K\} \subseteq \mathcal{V}\times\mathcal{V}$ a set of edges defined as nodal pairs. In a single sample of a graph signal, an input vector x_n from the graph input $\mathbf{x} \in \mathbb{R}^N$ is assigned to each node in $\mathcal{V}$. The graph structure, or topology of the graph, is represented in matrix form, typically by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, defined as:

$$A_{i,j} = \begin{cases} 1, & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The generally accepted graph network model proposed by Battaglia et al. (2018) projects the nodes, edges, and possibly the entire graph to a d-dimensional space. The nodal embedded states iteratively add information from their k-hop neighborhood over numerous message-passing steps, in which nodes are updated as a function of the embedded values of neighboring nodes and their own previous state. Information is propagated between nodes, edges and possibly the whole graph (Gilmer et al., 2017). The broadcasting of signals on the graph is computed as a series of local operations, commonly matrix multiplications with adjacency matrices are employed. Taking $z_n^{(l)} \in \mathbb{R}^d$ as the embedded state of node n on iteration l, the embedding, updating and outputting steps are:

$$\mathbf{z}^{(0)} = \mathbf{x} \tag{6}$$

$$\mathbf{z}^{(l+1)} = f\left(\mathbf{z}^{(l)}, \bar{\mathbf{A}}\right) \tag{7}$$

$$\mathbf{z}^{(L)} = \mathbf{y} \tag{8}$$

where $\mathbf{z}$ represents the embedded state of the graph, $\mathbf{y}$ the nodal prediction output, f a non-linear state update function that depends on the accumulated messages of neighboring nodes and the previous state of the nodes themselves; $\bar{\mathbf{A}} = \mathbf{A} + \mathcal{I}$, with $\mathcal{I}$ as the identity matrix, is an adjacency matrix augmented with self loops (to allow the incorporation of information from the nodes themselves, and not just from neighboring nodes) (Kipf and Welling, 2017).

The elements of the power grid are related through the grid topology, and the state of the grid depends on both external injections and this configuration, it is thus evident that a natural correspondence between power grids and graph networks exists. However, the defined bus types are fundamentally different in that each has different characteristics and a different number of expected outputs. Furthermore, the branch characteristics are important factors in determining the resulting voltages, such that they cannot be treated as simple edges, but are not expected to produce an output. In essence, the different grid elements should be treated as distinct node types, as shown in Fig. 1(b).

TGNs are a generalization of GNNs, in that instead of defining nodes and edges as the graph elements, the concept of node types is adopted. This way the edges can be defined as a type of node, and a finite number of additional node types can be designated. This outlook presents several advantages; each node type can have different output, projected state dimensions, and update function parameters.

This way a relevant difference with respect to conventional GNNs is the way the message-passing and update iterations are computed. For a TGN with a ζ set of node types, message passing functions are defined for node types that have at least one pair of adjacent nodes, to compute messages from one projection dimension to the other:

$$\mathcal{M} = \{\mu : \mathbb{R}^{d_j} \to \mathbb{R}^{d_i} \mid i, j \in \zeta,\ A_{i,j} \neq \mathbf{0}\} \tag{9}$$

The outputs from the message passing functions are propagated via matrix multiplication with adjacency matrices that indicate the sparsity pattern between the different node types, e.g. an adjacency matrix between types i and j nodes is represented as $A_{i,j} \in \mathbb{R}^{d_i \times d_j}$. For each node type i, their corresponding update function, ϕ_i concatenates their previous state with the propagated messages from neighboring node types, such that:

$$\phi_i : \mathbb{R}^{d_i + D_i} \to \mathbb{R}^{d_i} \tag{10}$$

$$D_i = \sum_{\substack{\mu:\mathbb{R}^{d_j} \to \mathbb{R}^{d_i} \\ j \in \mathcal{N}(i)}} d_i \tag{11}$$

where $\mathcal{N}(i)$ represents the set of all node types that directly interact with type i nodes, and D_i is the dimension of the final accumulation of messages that are concatenated from the neighboring node types. Thus the update function input for each node consists of the final accumulation of messages concatenated with their previous node state.

This way, the TGN structure takes as input a feature vector for every node of every node type, i.e. $\mathbf{x}_i \in \mathbb{R}^{f_i}$, for a node type i, where f_i is the dimension of the input features. All nodes are projected to their corresponding embedding space through a linear function, $\gamma_i : \mathbf{x}_i \to \mathbf{z}_i \in \mathbb{R}^{d_i}$. Then message passing and node update functions are repeated iteratively for L message passing steps. The nodal outputs are obtained from a final mapping, $\varphi_i : \mathbf{z}_i \to \mathbf{y}_i \in \mathbb{R}^{o_i}$; where o_i is the desired output dimension for i type nodes.

3. Methodology

In this section a description of how the TGN framework is applied to solve the power flow problem is presented. It is shown that by training four small MLPs, the power flow problem can be solved in linear time, robust to changes in topology (in particular to single branch line outages) and different branch characteristics. The training is not supervised, but physics-informed, and the solver architecture is modular in nature. First a description of the proposed TGN based model is reported, and then the training mechanism is explained.

3.1. TGN based power flow solver

In the proposed method, a predefined number T of TGN layers is used to iteratively approximate the missing voltage values at every node of the power grid. The message passing and update functions, μ and ϕ, are small fully connected neural networks; their parameters are the only ones that must be learned. These functions are the same for all nodes of the same type in the same layer, supporting the concept of combinatorial generalization, this way, the same TGN architecture can operate with input graphs of different sizes and shapes.

While the embedding, message passing, update and output functions of each TGN layer are independently parameterized, the layers are structurally the same; each one is defined by four types of nodes: branch nodes ($\mathcal{V}_E$) to include branch characteristics, PV nodes ($\mathcal{V}_{PV}$) for generator buses, PQ nodes ($\mathcal{V}_{PQ}$) for load buses, and slack nodes ($\mathcal{V}_S$) for slack buses. Different input features are defined for each type of node, as shown below for a layer t:

$$\mathbf{x}_{PV}\begin{cases} V_i(t) \\ \theta_i(t) \\ \Delta P_i(t) \\ Qg_i(t), \end{cases} \quad \mathbf{x}_{PQ}\begin{cases} V_j(t) \\ \theta_j(t) \\ \Delta P_j(t) \\ \Delta Q_j(t), \end{cases} \quad \mathbf{x}_{S}\begin{cases} V_s \\ \theta_s \\ Pg_s(t) \\ Qg_s(t), \end{cases} \quad \mathbf{x}_{E}\begin{cases} \rho(r_k, x_k) \\ \delta(r_k, x_k) \\ b_k \end{cases} \tag{12}$$

$\forall i \in \mathcal{V}_{PV}, \forall j \in \mathcal{V}_{PQ}, \forall s \in \mathcal{V}_S, \forall k \in \mathcal{V}_E$

As was mentioned in Section 2.1, the branch series impedance is given by the complex value: $z_k = r_k + ix_k = |z_k| \angle\varphi$, however, making reference to the conventional admittance matrix, the branches admittance magnitude and phase are chosen as features. The admittance is given by: $y_k = \frac{1}{z_k} = \frac{1}{|z_k|}\angle -\varphi = \rho\angle\delta$. Defining the magnitude and phase of the series admittance of each branch as ρ and δ, respectively, the first two branch node features are thus established as:

$$\rho(r_k, x_k) = \frac{1}{\sqrt{r_k^2 + x_k^2}} \tag{13}$$

$$\delta(r_k, x_k) = -\arctan(\frac{x_k}{r_k}) \tag{14}$$

The last branch node feature b_k corresponds to the total line charging susceptance. These branch input features remain unchanged throughout all TGN layers. Concerning the other features, the rotating angle of the buses is measured relative to the chosen slack bus, as is common in power flow analyses, i.e. $\theta_n = \theta_s,\ \forall n \in (\mathcal{V}_{PV} \bigcup \mathcal{V}_{PQ})$. Apart from this, an initial 'flat' guess of voltage magnitude values is established, with $V_j = 1$ (per unit), $\forall j \in \mathcal{V}_{PQ}$. This initial voltage state is used for the first TGN layer, and the following TGN layers receive the voltage approximation of the previous layer; after each approximation, the power balance error (ΔP and ΔQ) is calculated at each bus and is used as part of the input features for the next TGN layer. The reactive power at generator buses is locally compensated, such that $\Delta Q_j = 0$, this does not provide additional information and is thus excluded from the $\mathcal{V}_{PV}$ input features. A similar situation happens with the slack node where both active and reactive powers are compensated; thus, the calculated powers generations are used as input features. Furthermore, the same adjacency matrices are used for every layer of the TGN based solver since the configuration of the power grid is not altered between layers.

For the message-passing steps three distinct adjacency matrices are defined: between $\mathcal{V}_E$ nodes and $\mathcal{V}_{PV}$, $\mathcal{V}_{PQ}$ and $\mathcal{V}_S$ nodes, as shown in Eq. (15). Only three adjacency matrices are needed because bus type nodes cannot be directly connected to each other, but instead are always connected through branch type nodes. The adjacency matrices may be transposed depending on whether information is moving from a branch node to a bus node, or vice versa.

$$\begin{aligned} &\mathbf{A}_{PV,\ E} \in \mathbb{R}^{|\mathcal{V}_{PV}| \times |\mathcal{V}_E|} \\ &\mathbf{A}_{PQ,\ E} \in \mathbb{R}^{|\mathcal{V}_{PQ}| \times |\mathcal{V}_E|} \\ &\mathbf{A}_{S,\ E} \in \mathbb{R}^{1 \times |\mathcal{V}_E|} \end{aligned} \tag{15}$$

A predefined number L of message passing and state update steps is set; for this procedure, the same NNs μ and ϕ, are applied at each iteration to propagate and aggregate information across the graph, and update the graph state. At the final update step, each node has shared information with neighboring nodes L−hops away (Battaglia et al., 2018). This iterative message-passing and updating process is illustrated in Fig. 2. The output of each TGN layer is obtained by decoding the final states of the $\mathcal{V}_{PV}$ and $\mathcal{V}_{PQ}$ node types. The layer outputs correspond to the inferred change in voltage values, ΔV and $\Delta\theta$, with respect to the input voltage values, for the corresponding TGN layer, as shown below:

$$\mathbf{y}_{PV}\left\{ \Delta\theta_i \right. \quad , \quad \mathbf{y}_{PQ}\begin{cases} \Delta V_j \\ \Delta\theta_j \end{cases} \quad , \tag{16}$$

$\forall i \in \mathcal{V}_{PV}, \forall j \in \mathcal{V}_{PQ}$

The voltage values after every TGN layer are updated by:

$$\hat{V}(t+1) = \Delta V + \hat{V}(t) \tag{17}$$

$$\hat{\theta}(t+1) = \Delta\theta + \hat{\theta}(t) \tag{18}$$

With the updated voltage values, Eqs. (3) and (4) are used to calculate the power balance error in each bus, the reactive power

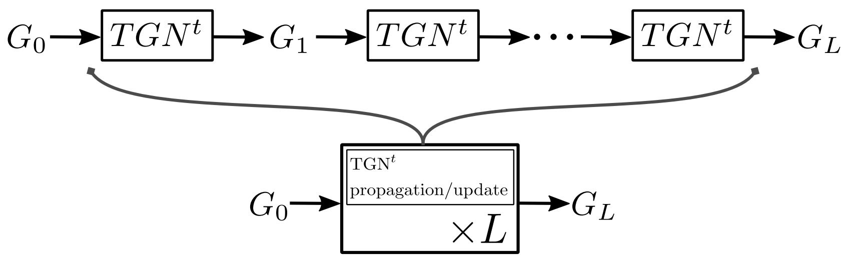


Fig. 2. Message passing and update state procedure (shared weights).

compensation in generator buses, and both active and reactive power compensation in the slack bus. If the last layer has not been reached, the voltage values and power balance error values are used to determine the graph input for the next TGN layer, otherwise, they are used to evaluate the cost function. The iterative portion of the proposed power solver structure is summarized by Algorithm 1 and illustrated in Fig. 3. If the model is being trained, the final voltage inference is used with the power equilibrium equations to calculate the loss function and then the Adam optimization algorithm is applied to find new NN parameters, as is explained in Section 3.3. If the model is not being trained, the process ends with the final voltage inference values.

Algorithm 1 TGN based PF solver

```
Require: $\mathcal{G} = \bigcup_{i\in\zeta} \mathcal{V}_i, \quad A_{i,j} \in \mathbb{R}^{d_i \times d_j} | i,j \in \zeta, \quad \zeta = \{PV, PQ, S, E\}$
      Input: $\mathcal{S}$ = grid state (injections, branch characteristics, topology)
 1: $\hat{V}^{(0)}_{PQ} = 1$, $\hat{\theta}^{(0)}_{PV,PQ} = 0$    ▷ Flat start
 2: $t = 0$
 3: while $t \le T$ do    ▷ Iterate over a predefined number of TGN layers
 4:     $\Delta P^{(t)} = P_g - P_d - P_{bus}(\hat{V}^{(t)}, \hat{\theta}^{(t)})$    ▷ Power balance error
 5:     $\Delta Q^{(t)} = Q_g - Q_d - Q_{bus}(\hat{V}^{(t)}, \hat{\theta}^{(t)})$
 6:     for each $i \in \zeta$ do    ▷ For all node types
 7:         $\mathbf{x}_i = \mathbf{f}_i(\hat{V}^{(t)}, \hat{\theta}^{(t)}, \Delta P^{(t)}, \Delta Q^{(t)}, \mathcal{S})$    ▷ Input features: $\mathbf{x}_i \in \mathbb{R}^{N_i \times f_i}$
 8:         $\mathbf{z}^{(0)}_i = \gamma_i(\mathbf{x}_i)$    ▷ Node embedding: $\mathbf{z}_i \in \mathbb{R}^{N_i \times d_i}$
 9:         $l = 0$
10:         while $l \le L$ do    ▷ $L$ message-passing steps
11:             $\bar{\mu}_i = \text{concat}\left(A_{i,j} \cdot \mu_i\left(\mathbf{z}^{(l)}_j\right)\right), \forall j \in \mathcal{N}(i)$    ▷ Message aggregation
12:             $\mathbf{z}^{(l+1)}_i = \phi_i\left(\mathbf{z}^{(l)}_i, \bar{\mu}_i\right)$    ▷ Update $\forall i \ne S$
13:             $l \leftarrow l + 1$
14:         end while
15:     end for
16:     $\hat{V}^{(t)}_{PV} = \varphi_{PV}\left(\mathbf{z}^{(L)}_{PV}\right)$    ▷ Outputs: $\mathbf{y}_i \in \mathbb{R}^{N_i \times o_i}$
17:     $\hat{V}^{(t)}_{PQ}, \hat{\theta}^{(t)}_{PQ} = \varphi_{PQ}\left(\mathbf{z}^{(L)}_{PQ}\right)$
        $t \leftarrow t + 1$
18: end while
      Output: $\hat{\theta}^{(T)}_{PV}, \hat{V}^{(T)}_{PQ}, \hat{\theta}^{(T)}_{PQ}$    ▷ Final voltage inference
```

3.2. Model computational complexity

Each TGN layer is composed of four main functions: encoding, message passing, updating and decoding. The four functions are defined by small MLPs, with either one or two layers. All MLPs with two layers have one layer with a hyperbolic tangent activation function, and a linear layer; the MLPs with a single layer are linear. Different instances of the encoding MLPs are defined for each of the four node types ($\mathcal{V}_{PV}$, $\mathcal{V}_{PQ}$, $\mathcal{V}_E$, $\mathcal{V}_S$); these MLPs have a single layer of size $f_x \times d_x$ for each node type x, where f_x and d_x are the corresponding feature vector size and embedding dimension, respectively. Five message passing functions are defined: two to exchange information to and from $\mathcal{V}_{PV}$ and $\mathcal{V}_E$, two to exchange information between $\mathcal{V}_{PQ}$ and $\mathcal{V}_E$, and one for passing information from $\mathcal{V}_S$ to adjacent $\mathcal{V}_E$ nodes. The message passing MLPs have two layers, the first of size $d_{IN} \times d_{IN}$, and the second of size $d_{IN} \times d_{OUT}$. When a type x node casts information unto a type y node ($\mathcal{V}_x \to \mathcal{V}_y$), d_{IN} and d_{OUT} represent the embedding size of the type x and type y nodes, respectively. Only $\mathcal{V}_{PV}$, $\mathcal{V}_{PQ}$ and $\mathcal{V}_E$ are updated; $\mathcal{V}_S$ nodes pass their corresponding information through the $\mathcal{V}_E$ nodes but do not need to be updated, as no information is sent to them and no output is required from them. $\mathcal{V}_E$ nodes receive information from $\mathcal{V}_S$, $\mathcal{V}_{PV}$ and $\mathcal{V}_{PQ}$; $\mathcal{V}_{PV}$ and $\mathcal{V}_{PQ}$ nodes only receive information from $\mathcal{V}_E$. The aggregating and update function for $\mathcal{V}_E$ has two layers, one of size $(d_E + d_{PV} + d_{PQ} + d_S) \times d_E$, and the other of size $d_E \times d_E$. The update functions for generator and load bus type nodes also have two layers, one of size $(d_x + d_E) \times d_x$, and the other of size: $d_x \times d_x$, where $x = PV$ or $x = PQ$. Only $\mathcal{V}_{PV}$ and $\mathcal{V}_{PQ}$ require an output, their decode MLPs consist of a single layer of size $d_x \times o_x$, where $o_{PV} = 1$ and $o_{PQ} = 2$. In total, each TGN layer has the following number of trainable parameters P:

$$
\begin{aligned}
P = {} & (4d_{PV} + 4d_{PQ} + 4d_E + 2d_S)d_E + 3d_{PV}^2 + 3d_{PQ}^2 + d_S^2 \\
& + (F_{PV} + G_{PV})d_{PV} + (F_{PQ} + G_{PQ})d_{PQ} + F_E d_E + F_S d_S
\end{aligned} \tag{19}
$$

If all embedding dimensions are the same d value, then:

$$
P = 21d^2 + (F_{PV} + G_{PV})d + (F_{PQ} + G_{PQ})d + F_E d + F_S d \tag{20}
$$

In this particular case, for simplicity, all nodes were embedded to the same dimension, with $d = 16$, so that $P = 5648$. This value is independent of the size of the electrical grid.

With N_x representing the cardinality of a type x node set, the encoding function for a type x node has complexity $\mathcal{O}(f_x N_x d_x)$, and since f_x and d_x are predefined constants, this translates to a $\mathcal{O}(N_x)$ complexity. Similarly, each decoding and message passing function has complexity $\mathcal{O}(N_x)$. The update function includes an aggregation procedure which involves the multiplication of sparse adjacency matrices, with dense matrices that represent messages passed from one type of node to another. The total amount of values in all sparse matrices is $2N_E$, the complexity of the aggregation multiplications for a type x node is at most $\mathcal{O}(2N_E d_x)$. The total amount of nodes $N = N_{PV} + N_{PQ} + N_S + N_E$ depends on the particular case of electrical grid, but as all operations have at most $\mathcal{O}(DN) = \mathcal{O}(N)$ complexity (with D being some constant dependent on the chosen hyperparameters), the time complexity of the proposed solver is linear with respect to the size of the electrical grid.

3.3. Model training

The proposed TGN solver is trained in batches, the independent NNs of the TGN layers are trained simultaneously based on a cost function that only considers the final voltage inference and the resulting power balance error. The cost function is given by:

$$
\frac{1}{H}\sum_{h=1}^{H}\left(\frac{1}{N}\sum_{n=1}^{N}\left(\Delta P_{n,h}^2 + \Delta Q_{n,h}^2\right)\right) \tag{21}
$$

where N represents the total number of buses, and H the total number of samples (n and h being the node and sample indices, respectively). The gradients of this cost function with respect to the parameters of the

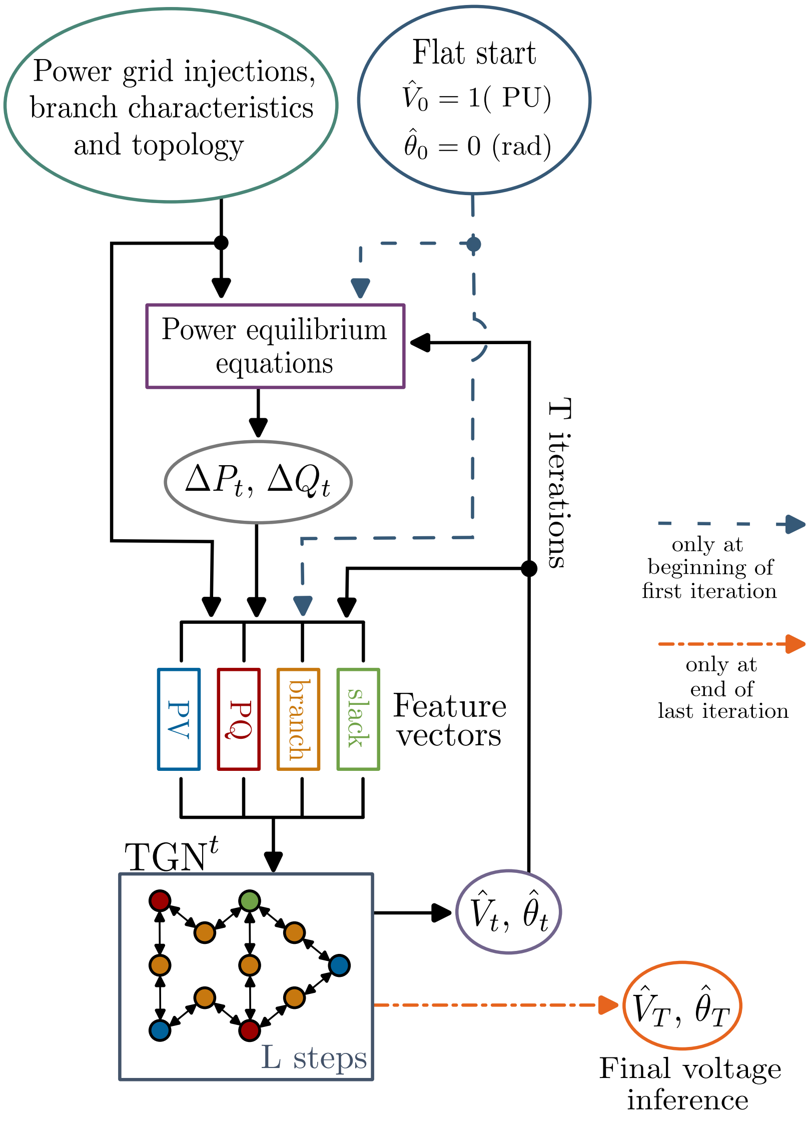

Power grid injections, branch characteristics and topology
Flat start
$\hat{V}_0 = 1$(PU)
$\hat{\theta}_0 = 0$ (rad)
Power equilibrium equations
ΔP_t, ΔQ_t
T iterations
only at beginning of first iteration
only at end of last iteration
PV
PQ
branch
slack
Feature vectors
TGN^t
L steps
$\hat{V}_t$, $\hat{\theta}_t$
$\hat{V}_T$, $\hat{\theta}_T$
Final voltage inference

Fig. 3. TGN based power solver structure.

embedding, message-passing, update and output NNs is calculated with the backpropagation algorithm. These gradients are used to modify the values of the NN parameters via the Adam optimization algorithm, afterwards a new batch of data is introduced to the TGN model and the process is repeated until the cost function converges to a minimum value. This way, as is evident, the learning is unsupervised; the goal of the cost function is to enforce Kirchhoff's current law by minimizing the power balance error in all nodes.

4. Numerical tests and discussion

In this section the design of the experiments performed to validate the proposed TGN based power flow solver is described, along with the results obtained.

4.1. Dataset

The data for all experiments is based on benchmark IEEE test cases, similar to the default test cases available for Matpower (Zimmerman et al., 2011); perturbations are added to the injections, branch characteristics and grid topology for each sample.

The information of the IEEE test cases is imported through three two dimensional arrays, with information for the buses, generators, and branches, respectively. From the bus array, the values of the active and reactive power demand are extracted, and a proper ID number is assigned to each substation. Uniform noise is added to the value of the active and reactive power loads, Pd and Qd, so that the resulting values vary between 50% and 150% of the original value. The power load is restricted so that the rare case of the total load demand being higher than the sum of the maximum power generation limits is avoided. From the branch array, the indices of the buses at each side of the branch is collected, as well as the resistance, reactance, susceptance and tap ratio values of the branch. Similarly as with the bus perturbations, uniform noise is added to these values so that they are between 90% and 110% of the test case value. An important aspect of the presented work is the introduction of change to the electrical grid topology in the different samples during training of the system; to this effect, in each sample a different random branch is disconnected. This way both injection and topology changes are involved during training. From the generator array, the ID of the generator bus is considered, along with the maximum permitted active power generation and the nominal voltage magnitude of these buses. The voltage magnitude is uniformly sampled between 90% and 110% of the nominal value, the active power generation is uniformly sampled between 25% and 75% of the allowed range. All voltage magnitude values from load buses are initialized to 1 P.U. and all voltage phase values are set to the slack bus angle reference.

This way, a group of objects that represent a batch of electrical grid samples is formed. Said representation is structured so that it is ready to be used to calculate adjacency matrices and the inputs to the proposed TGN based model.

4.2. Model construction

In the following tests, three electrical grid sizes are employed based on the case 30, case 57 and case 118 standard IEEE power grids. For this reason, three instances of the proposed solver are generated, each trained on an electrical grid of fixed size. The three instances of the TGN based power solver share the same hyperparameters: number of TGN layers, number of message passing and update steps, embedded dimension of node types, and learning rate. The number of TGN layers is empirically chosen to produce a precise enough solution while maintaining the overall size of the solver relatively small; this number is set at $T = 15$. Furthermore, to avoid the distortion of messages being propagated from distant nodes (Topping et al., 2022), only two message passing and update steps are defined ($L = 2$) for each TGN layer. As once mentioned previously, the embedded dimension for all types of nodes is set to $d = 16$. The optimizer used is the Tensorflow (Abadi et al., 2015) implementation of the Adam algorithm, with a learning rate of $1e-4$, and all other parameters are left with the default values. The three instances are trained relatively quickly, with only 1250, 1500 and 1500 learning iterations for the TGN instances trained on case 30, case 57 and case 118 grids, respectively.

4.3. Conventional test

The first experiment consists of testing the three instances of the proposed solver on electrical grids of the same size as the ones they are trained on, changing power grid injections and disconnecting random branches. For each grid size, 20 power grid states are generated; the corresponding TGN instance is applied to infer the missing voltage values at each node. To validate the obtained results, they are compared with the solutions calculated with the trusted and conventional Newton–Raphson (N–R) method using Matpower (Zimmerman et al., 2011). Because of the way the batch samples are generated, some input samples result in non-feasible grid states that do not converge with the N–R method, for these cases the proposed solver does infer some solution, but in the presented results only those that converged with the N–R method are considered.

To further explain the inference procedure, Fig. 4 shows the absolute difference between the final N–R based result and the outputs obtained at distinct TGN iterations of the proposed method, for a single sample of the case 118 grid. This way, *iter* 0 corresponds to the output of the first TGN layer; *iter t* represents an intermediate layer, in this case $t = 7$; *iter T* represents the final output, with $T = 15$. The abundance of low error vertices on the magnitude side of the first layer is due to the quantity of generator (PV) nodes, for which the voltage magnitude is known from the original state of the grid. As the iterations advance, it is shown that the output of each node approaches the N–R output, even though the learning is not supervised and the N–R result is not known during training.

To exemplify the similarity between the resulting voltage values obtained with the N–R method and the proposed TGN based solver Figs. 5 to 7 illustrate the final voltage magnitude and phase results of a single sample, obtained with the proposed TGN based solver and the N–R method. Fig. 5 shows the results for a sample state of case 30 grids, displaying the voltage magnitude and phase of each substation on the left and right graphs, respectively. Figs. 6 and 7 show similar experiments for case 57 and case 118 grids, respectively. These are purely illustrative graphs, since they only show the outputs of a single power grid instance.

In order to show a more general perspective of the precision of the proposed solver, Figs. 8 to 10 show scatter plots with the N–R based solutions and the proposed solver solutions, for test batches with 20 samples. The value of the correlation coefficient between the two results is also shown in the bottom right of the scatter plots. Fig. 8 shows the correlation between the proposed solver solution and the N–R based solution, for magnitude (PU) on the left and for phase on the right (radians). That is, the N–R based solutions, for all the buses in all the samples of the test batch, are compared to the corresponding values obtained with the proposed method. Figs. 9 and 10 show similar plots for case 57 and case 118, respectively. In all cases, there is a high level of correlation with all coefficients being above 0.98; the least correlated case is the one corresponding to the voltage phase solution of the case 118 grids. The most correlated case corresponds to the voltage magnitude solution of the case 118 grids, although this can be partly explained by the high rate of generator nodes in that specific case (the voltage magnitude values are known beforehand).

Table 3
Voltage magnitude RMSE.

	Test size		
	Case 30	Case 57	Case 118
DC	0.040139	0.045667	0.028195
TGN Trained on case 30	**2.5092e–04**	3.9902e–03	1.7344e–03
TGN Trained on case 57	8.5203e–03	**2.3706e–03**	2.0319e–03
TGN Trained on case 118	3.6260e–04	3.9346e–03	**2.3699e–04**

Table 4
Voltage phase RMSE.

	Test size		
	Case 30	Case 57	Case 118
DC	0.012974	0.016247	0.079729
TGN Trained on case 30	**2.2852e–03**	0.068866	0.017480
TGN Trained on case 57	0.01229	**4.4389e–03**	**1.2320e–03**
TGN Trained on case 118	6.1418e–03	0.01686	0.010423

4.4. Extrapolation to different grid sizes

One of the crucial points of the proposed model is the capability to be tested on grids of different size from the ones they are trained on, due to the graph structure representation of the system. To evaluate the generalization performance of the proposed solver when faced with different grid sizes, each of the three TGN instances is tested on batches of the other two grid sizes that do not correspond to the ones they were trained on. Each grid in the test batch is obtained as described in Section 4.3, with varying injections, line characteristics and grid configuration through the elimination of a random branch.

To add another point of comparison to the results, the test samples are additionally solved using the DC approximation method, which does not consider the reactive power in the electrical grid and uses linear network equations that relate real power to bus voltage angles (instead of complex bus voltages). The DC approximation is simple and robust, and for these reasons, sometimes used for contingency or real-time dispatch analyses (Van Hertem et al., 2006).

In Tables 3 and 4, the solutions obtained with the proposed method and the DC approximation are compared to the results obtained with the N–R method, which is considered the correct solution. The comparison is measured using the root mean square error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{H}\left(\sum_{j=1}^{N}(x_{i,j} - \hat{x}_{i,j})^2\right)}{NH}} \tag{22}$$

where $x_{i,j}$ represents the solution obtained with the newton raphson method for a bus j at test sample i, $\hat{x}_{i,j}$ represents the corresponding solution obtained with either the DC approximation method or the proposed TGN based method, N represents the total number of buses of the tested grid and H represents the total number of samples in the test batch.

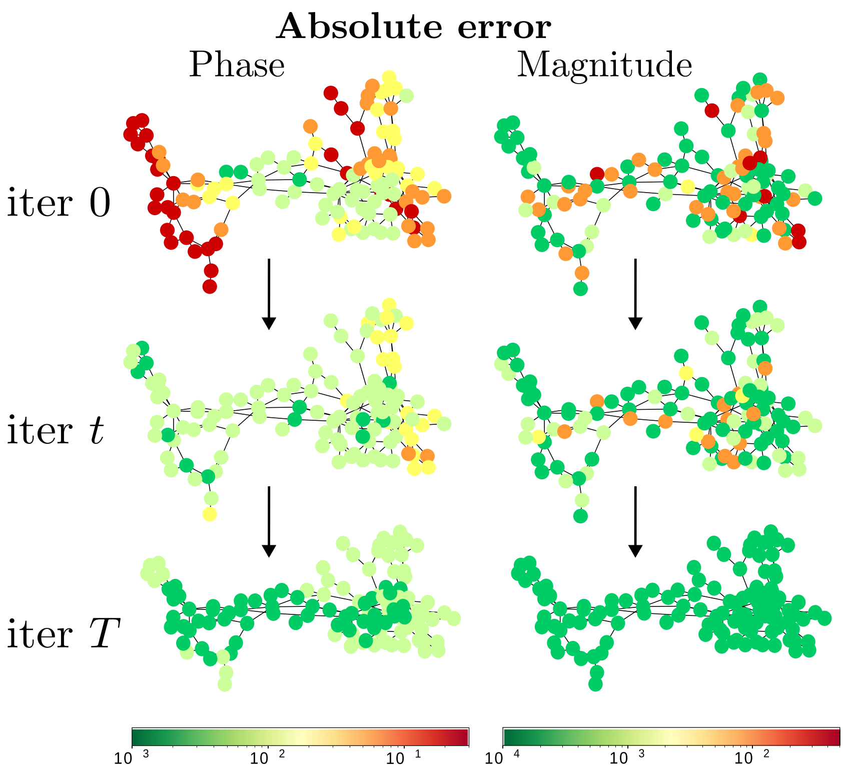


Fig. 4. Evolution of the absolute difference between the proposed method outputs throughout the TGN layers and the final N–R based output, for a single sample of the case 118 grid, showing voltage phase difference in radians (left) and voltage magnitude in P.U. (right).

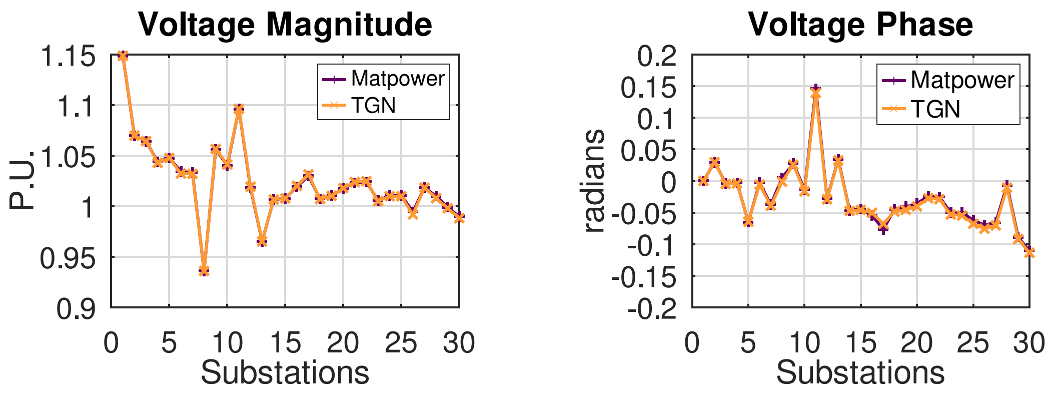


Fig. 5. Voltage magnitude and phase solutions for a single sample of the case 30 grid.

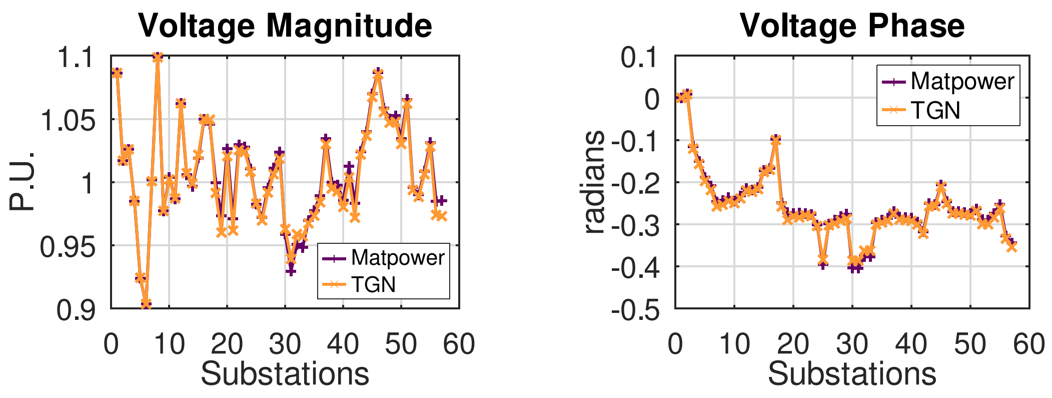


Fig. 6. Voltage magnitude and phase solutions for a single sample of the case 57 grid.

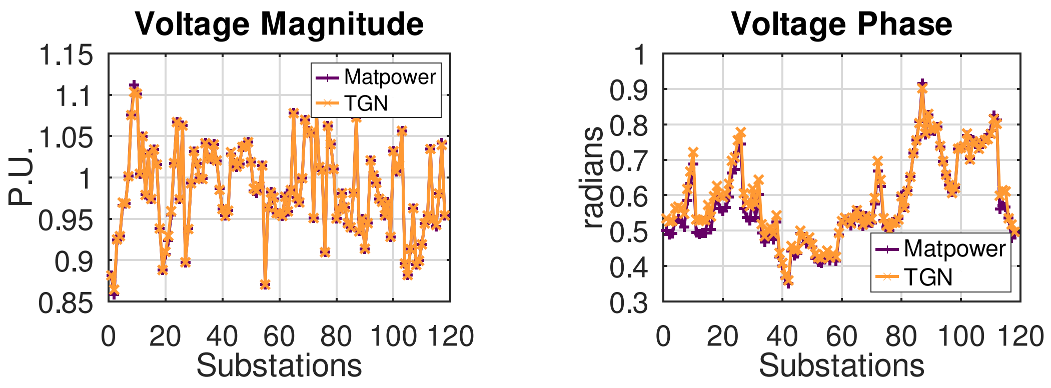


Fig. 7. Voltage magnitude and phase solutions for a single sample of the case 118 grid.

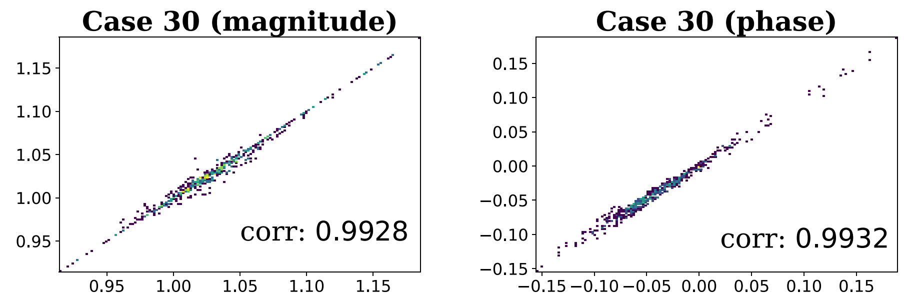


Fig. 8. Correlation of TGN based power solver with Matpower's Newton Raphson solution for case 30 V magnitude (left) and phase (right).

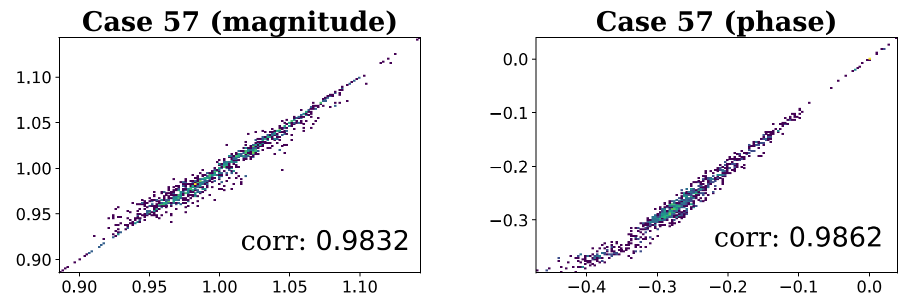


Fig. 9. Correlation of TGN based power solver with Matpower's Newton Raphson solution for case 57 V magnitude (left) and phase (right).

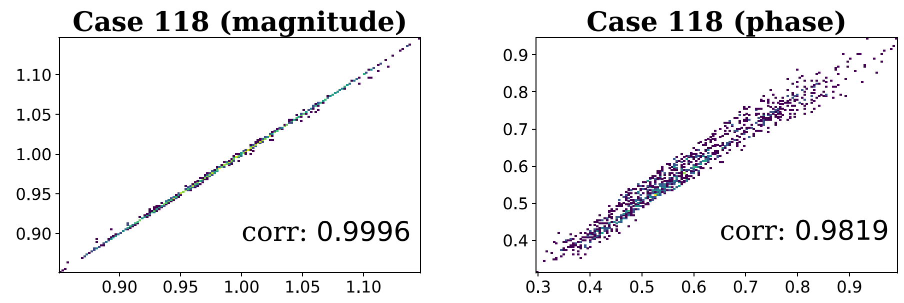


Fig. 10. Correlation of TGN based power solver with Matpower's Newton Raphson solution for case 118 V magnitude (left) and phase (right).

The first row of Table 3 shows the voltage magnitude RMSE of the DC approximation results and the N–R method results, tested on grids from the three different test case sizes. The following rows show the RMSE of the solutions obtained with different instances of the proposed method, trained on case 30, case 57 and case 118 power grids, and each of them tested on all grid sizes. Table 4 is similar, but comparing voltage phase values.

The best results for each test case are marked in bold; as would be expected, most of the best results are obtained from grids tested on grids of the same size as the ones they were trained on. The results of the TGN instances tested on grids of the same size as the ones they are trained on are the same as in Section 4.3. As shown in Table 3, even when tested on grids of different size, the proposed method performs better than the DC approximation method when calculating voltage magnitude, this is expected as the DC approximation method considers all voltage magnitudes constant at 1 PU. Table 4 shows that in most cases, the proposed TGN based method performs better than the DC approximation when calculating voltage phase as well. At times the

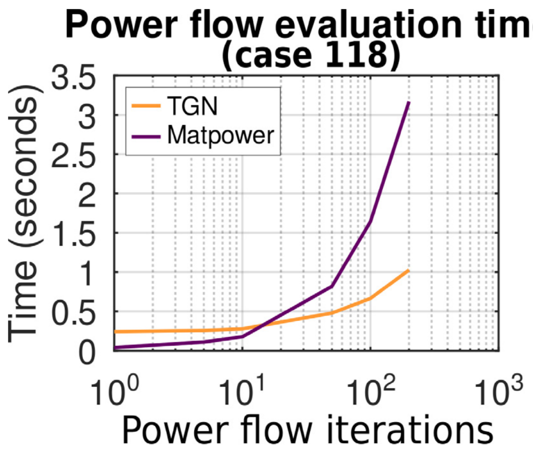


Fig. 11. Time in seconds needed to solve different numbers of power grid scenarios, for the case of 118 buses, using the TGN based solver and the N–R based Matpower approach.

results are very similar, like with the TGN instance trained on case 57 and tested on case 30 grids, and with the TGN instance trained on case 118 but tested on case 57 grids. In the worst case, which is with the TGN trained on case 30 but tested on case 57 grids, the DC approximation obtained a smaller error, however the TGN solution is still quite decent. As shown, the worst results are obtained when testing the case 57 grid, this can be explained by the wide range of values the injection parameters can take for this particular grid case, which presents values very different from the ones seen in the other two cases. Still, the errors in the worse case are considerably small, and it is shown that smaller grids are able to generalize to bigger grids, and vice versa.

4.5. Time considerations

As was mentioned in Section 3.2, the time complexity of the model is linear, and thus the calculation time does not increase as sharply as the N–R method with respect to the size of the electrical grid. To illustrate this, Fig. 11 shows the time in seconds needed to run a different number of scenarios of the 118 bus test case power grid. The number of power flow iterations is shown on a logarithmic scale. In this case, running n scenarios multiplies the number of inputs by n for the proposed solver, and the scenarios are solved in parallel. With the N–R based approach, the n scenarios are processed sequentially. It is shown that although for a small number of inputs the N–R based method is faster, as the number of inputs grows, the time required is substantially shorter with the proposed method. These considerations show an important reduction in the time needed to carry out numerous power flow iterations, which in combination with the robustness to single branch outages and differences in branch characteristics, make it a beneficial tool for power grid planning and risk assessment.

5. Conclusion

The proposed TGN based power flow solver takes advantage of the intuitive connection between the electrical grid data and graph representations to learn the relationships and dynamics between the different types of elements present in electrical grid models to analyze power flow. An important aspect of the presented work is the generalization capability to infer decent results for essentially different grids (varying injection, branch characteristics and topology). As far as the authors know, there is no other work that solves the power flow problem by training neural networks with both different injections and different grid configurations. The proposed method does not imitate any other existing method, but rather is based on minimizing the active and reactive power imbalance at each node of each sample during the training of the parameters.

The proposed method exploits several benefits of GNNs, e.g. they scale linearly with the number of edges and embedding size. Furthermore, since the voltage variables are not directly modified by the proposed method, we completely avoid the computation of Jacobian matrices and their inverse, which is necessary in the N–R method. These two characteristics are key reasons as to why the proposed model computation time scales in a more linear way with respect to the test grid size than the N–R method. It is worth mentioning that the testing of different grid sizes is not possible for conventional MLP methods, and that these methods are inefficient for larger grids as their size depends on the grid size, whereas the presented method does not, due to it being based on local operations and shared modules.

Through the shown numerical tests, it is shown that the proposed TGN based method obtains results very close to those obtained with a conventional Newton–Raphson based method, even when the learning is unsupervised. The tests are carried out in batches, with each sample of the batch representing an independent electrical grid from the rest. However, in future work, steps can be taken to capture sequentiality in time, i.e. instead of the samples being autonomous from each other, if each sample represents the state of the grid over a certain time duration Δt, then H samples would represent the grid state evolution over a total time of $H \cdot \Delta t$ (assuming reliable injection forecast information is available). Additionally, the framework of the proposed model can be improved to continuously learn and adapt to the environment.

The results presented constitute a beneficial step toward a NN based system that can be applied for contingency analysis, real-time dispatch and techno-economic analyses, or as an aid to improve different stages of power system planning, optimization, operation and control of electrical grids.

CRediT authorship contribution statement

Tania B. Lopez-Garcia: Methodology, Software, Writing – original draft. **José A. Domínguez-Navarro:** Conceptualization, Supervision, Critical revision.

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.

Data availability

Data will be made available on request.

Acknowledgments

This paper was supported in part by the National Council of Science and Technology (CONACYT), Mexico, under scholarship 710033.

This paper was supported in part by the Spanish National Plan for Scientific and Technical Research and Innovation, under project PID2019-104711RB-I00.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. URL: https://www.tensorflow.org/. Software available from tensorflow.org.

van Amerongen, R.A., 1989. A general-purpose version of the fast decoupled loadflow. IEEE Trans. Power Syst. 4 (2), 760–770. http://dx.doi.org/10.1109/59.193851.

Avelar, P.H.C., Lemos, H., Prates, M.O.R., Gori, M., Lamb, L., 2019. Typed graph networks. ArXiv abs/1901.07984.

Babatunde, O., Munda, J., Hamam, Y., 2020. Power system flexibility: A review. Energy Rep. 6, 101–106. http://dx.doi.org/10.1016/j.egyr.2019.11.048, The 6th International Conference on Power and Energy Systems Engineering.

Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., Pascanu, R., 2018. Relational inductive biases, deep learning, and graph networks. arXiv.

Donon, B., Clément, R., Donnot, B., Marot, A., Guyon, I., Schoenauer, M., 2020. Neural networks for power flow: Graph neural solver. Electr. Power Syst. Res. 189, http://dx.doi.org/10.1016/j.epsr.2020.106547.

Fikri, M., Cheddadi, B., Sabri, O., Haidi, T., Abdelaziz, B., Majdoub, M., 2018. Power flow analysis by numerical techniques and artificial neural networks. In: 3rd Renewable Energies, Power Systems and Green Inclusive Economy, REPS and GIE 2018. Institute of Electrical and Electronics Engineers Inc., pp. 1–5. http://dx.doi.org/10.1109/REPSGIE.2018.8488870.

Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E., 2017. Neural Message Passing for Quantum Chemistry. In: JMLR.org (Ed.), International Conference on Machine Learning. Sydney, pp. 1263–1272. http://dx.doi.org/10.5555/3305381.3305512.

Glover, J.D., Sarma, M.S., Overbye, T.J., 2012. Power System Analysis and Design, fifth ed. Cengage Learning.

Hu, X., Hu, H., Verma, S., Zhang, Z.-L., 2021. Physics-Guided Deep Neural Networks for Power Flow Analysis. IEEE Trans. Power Syst. 36 (3), 2082–2092. http://dx.doi.org/10.1109/TPWRS.2020.3029557.

Kipf, T.N., Welling, M., 2017. Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net.

Lopez-Garcia, T.B., Coronado-Mendoza, A., Domínguez-Navarro, J.A., 2020. Artificial neural networks in microgrids: A review. Eng. Appl. Artif. Intell. 95, http://dx.doi.org/10.1016/j.engappai.2020.103894.

Owerko, D., Gama, F., Ribeiro, A., 2020. Optimal Power Flow Using Graph Neural Networks. In: IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, pp. 5930–5934. http://dx.doi.org/10.1109/ICASSP40776.2020.9053140.

Smith, O., Cattell, O., Farcot, E., O'Dea, R.D., Hopcraft, K.I., 2022. The effect of renewable energy incorporation on power grid stability and resilience. Sci. Adv. 9 (8), 760–770. http://dx.doi.org/10.1126/sciadv.abj6734.

Stott, B., 1974. Review of load-flow calculation methods. Proc. IEEE 62 (7), 916–929.

Topping, J., Giovanni, F.D., Chamberlain, B.P., Dong, X., Bronstein, M.M., 2022. Understanding over-squashing and bottlenecks on graphs via curvature. In: International Conference on Learning Representations. URL: https://openreview.net/forum?id=7UmjRGzp-A.

Tovar-Facio, J., Martín, M., Ponce-Ortega, J.M., 2021. Sustainable energy transition: modeling and optimization. Curr. Opin. Chem. Eng. 31, 100661. http://dx.doi.org/10.1016/j.coche.2020.100661.

Van Hertern, D., Verboornen, J., Purchala, K., Belrnans, R., KlingH, W.L., 2006. Usefulness of DC Power Flow for Active Power Flow Analysis with Flow Controlling Devices. In: The 8th IEE International Conference on AC and DC Power Transmission. IET, pp. 58–62. http://dx.doi.org/10.1049/cp:20060013.

Vankayala, V.S.S., Rao, N.D., 1993. Artificial neural networks and their applications to power systems—a bibliographical survey. Electr. Power Syst. Res. 28 (1), 67–79.

Xie, L., Singh, C., Mitter, S.K., Dahleh, M.A., Oren, S.S., 2021. Toward carbon-neutral electricity and mobility: Is the grid infrastructure ready? Joule 5 (8), 1908–1913. http://dx.doi.org/10.1016/J.JOULE.2021.06.011.

Zimmerman, R., Murillo-Sanchez, C., Thomas, R., 2011. MATPOWER: Steady-State Operations, Planning and Analysis Tools for Power Systems Research and Education,. Power Syst. IEEE Trans. 26 (1), 12–19. http://dx.doi.org/10.1109/TPWRS.2010.2051168.