# Graph Neural Network Power Flow Solver for Dynamical Electrical Networks

Tania B. Lopez-Garcia
*Electrical Engineering*
*University of Zaragoza*
Zaragoza, Spain
782726@unizar.es

José Antonio Domínguez-Navarro
*Electrical Engineering*
*University of Zaragoza*
Zaragoza, Spain
jadona@unizar.es

***Abstract*—This work presents a novel graph neural network (GNN) based power flow solver that focuses on electrical grids examined as dynamical networks. The proposed method employs a specific type of GNN that considers different types of nodes to allow the direct translation to the generator and load buses, and the branches present in power systems. The GNN model is trained with modified versions of IEEE test cases by permuting the configuration of the electrical grid for the distinct samples used during training. The training is carried out in an unsupervised manner, and the results are compared with a conventional and trustworthy Newton-Raphson based method. The presented results are shown to be accurate, and perform acceptably well even when tested on grids of different size from the ones they observed during training.**

***Index Terms*—power flow, graph neural networks, unsupervised learning**

## I. Introduction

In present times, the electrical grid on which modern society has becoming completely dependant, is becoming increasingly difficult to analyze and control [1]. This is due to the present-day challenges that have led to reforms in the conventional power grid, however, power flow analyses are still a critical step in the planning and operation of power systems [2]. To ensure that the grid remains in a secure state, many load flow analyses must be carried out, each corresponding to one of a variety of situations, such as different network configurations and grid injections [3]. The power flow analysis is a computational procedure with the purpose of finding the steady state operating characteristics of a power system network; it obtains the voltage phase and magnitude values at all buses in a power system, for a specified condition, i.e. topology, active and reactive power states for loads and generators [4]. With the voltage data known, the real and reactive power flow of each branch, as well as the reactive power of generators can be analytically determined with simple multiplication and addition operations.

Because of its nonlinear nature, for decades, a wide range of numerical analysis methods for solving nonlinear algebraic equations have been applied to solve the power flow problem [5], [6]. These methods are iterative in nature and many of them, such as the commonly applied Newton-Raphson method, require the computation of inverse matrices in each iteration. For large power networks this causes computational overhead, rendering these methods inadequate for certain instances of system monitoring and control [7]. Often, when fast and repetitive load flow estimations are required, the DC power flow method is used, which is significantly faster than the other AC power flow methods that do not neglect reactive power flows, but is also highly inaccurate.

For these reasons, there has been relatively recent interest in exploring new avenues based on heuristic methods, neural networks and fuzzy logic to deal with power systems subjects, such as the security assessment problem [8], [9], and others [10]–[12]. Given that for the power flow problem, one needs to repeatedly solve different instances which share patterns and characteristics , it is worth looking into data-driven algorithms and machine learning approaches which may exploit these patterns [13]. The hope is to develop new methods or complement existing numerical methods to improve their speed and capabilities for practical cases.

While works that implement neural networks to solve the power flow problem have been proposed since the early 90s [14], [15], they are not applicable to large, complex networks. Modern works in the area usually apply dense feed forward neural networks (FFNNs) that do not take advantage of the inherent graph structure and sparseness of the electrical grid [16], [17]. Additionally, most of them apply supervised learning to imitate conventional solvers, i.e. they require a large amount of training data to optimize the parameters of the model.

The work in this paper proposes a typed graph neural network (TGN) based system that abstracts the relationship between the generator and load buses to solve the deterministic steady-state power flow problem for dynamical networks, i.e. considering different grid configurations. Dynamical networks deal with elements that follow dynamical rules that include interactions with neighboring elements, with the underlying graph not being fixed, but allowing change [18]. The TGN is trained on essentially different electrical grids, with the same number of buses but different topology in each sample. This work focuses on modifying the structure of the power grid by simulating random line outages. The learning is not supervised, and focuses on minimizing the violation of the

This work is partly supported by the National Council of Science and Technology (CONACYT), scholarship number: 710033.

power equilibrium equations. The system is then tested on a grid of different size from the one it was trained on, and even in that case, acceptable results are obtained. The work in this paper presents an important step towards the final goal of developing a machine learning based system that is capable of generalizing to grids of different size, with varying topology and injections, to improve the speed and reduce the computational burden of essential power flow analyses; this in turn, would improve the static security assessment of large scale power systems, and bring other important contributions to the analysis of electrical grids.

## II. Methodology

The concepts of graph neural networks (GNNs) and TGNs are instrumental to the design of the proposed solution. An overview of GNNs and TGNs is presented in section II-A, followed by the proposed methodology in section II-B, which shows how TGNs are used to solve the power flow problem. Section II-C describes the training procedure.

### A. Preliminaries: Graph Neural Networks

Electrical grids can inherently be seen as graphs, one possible way to do this is by considering the buses as nodes and the branches as edges:

$$G = (\mathcal{V}, \mathcal{E}) \tag{1}$$

where $\mathcal{V}$ and $\mathcal{E}$ represent the set of nodes and edges, respectively.

Usually, each node in the GNN is assigned an input vector of features corresponding to it, represented by an input matrix:

$$\mathbf{x} \in \mathbb{R}^{N \times F_{in}} \tag{2}$$

where $N$ is the number of nodes and $F_{in}$ is the dimension of the input features. The topology of the GNN is defined by a graph shift operator (such as the adjacency or Laplacian matrices), which represent the graph structure in matrix form by indicating whether pairs of nodes are connected by an edge:

$$\mathbf{A} \in \mathbb{R}^{N \times N} \tag{3}$$

Since in this work the goal is to infer data at the nodes, the GNNs used in the proposed method compute a node level output:

$$\mathbf{y} \in \mathbb{R}^{N \times F_{out}} \tag{4}$$

where $F_{out}$ is the dimension of the desired output features. The following equations illustrate the encoding, update and decoding procedures. The input features of each node are embedded to a state of dimension $d$, as shown in (5), then each node is updated with the aggregated encoded features of neighboring nodes and the node itself for $T$ iterations, as shown in (6). Lastly, a linear output map is applied to obtain the desired output dimension in each node, as shown in (7).

$$\mathbf{z}(0) = \sigma\left(\mathbf{x}\mathbf{W}_{\sigma}(0)\right) \in \mathbb{R}^{N \times d} \tag{5}$$

$$\mathbf{z}(t+1) = \phi\left(\hat{\mathbf{A}}\mathbf{z}(t)\mathbf{W}_{\phi}(t)\right) \in \mathbb{R}^{N \times d},\ t = 1 \dots T-1 \tag{6}$$

$$\mathbf{y} = \mathbf{z}(T)\mathbf{W}_{y}(T) \in \mathbb{R}^{N \times F_{out}} \tag{7}$$

In the previous equations, each $\mathbf{W}(\cdot)$ represents a linear operator, $\sigma$ and $\phi$ are nonlinear activation functions for the embedding and update processes, respectively; $\hat{\mathbf{A}} = \mathbf{A} + \mathcal{I}$, where $\mathcal{I}$ represents the identity matrix, is an operator enforcing self loops in the graph to aggregate the features of each of the nodes as well as of the neighboring nodes.

By parameterizing the aggregation and update steps, the GNN is trained in an end-to-end fashion against a loss function. Thus, the GNN encodes critical graph structures that aid in solving the problem more efficiently than with simple FFNNs. Other reasons for using GNNs are described below.

By design, GNNs are invariant to permutation, which is a very useful feature in dealing with electrical grids since they have no unique representation, i.e. the buses can be renamed but the electrical grid is the same.

Graph representations of electrical grids, especially larger ones, are sparse (the number of edges is significantly lower than the possible number of edges). Due to the local nature of GNN operations (aggregations and updates), GNNs are sparsity aware and scale linearly with the number of edges and embedding size [19] .

TGNs present a way to generalize the concept of GNNs by expanding the possible types of elements of graphs to more than just nodes and edges [20]. TGNs simply consider all elements as different types of nodes, each type can have input features defined, adjacency matrices between them, and most importantly, their own aggregation and update functions. This means that each type of node can be embedded into a different dimension, which causes the aggregation and update steps to be separated into two individual operations, e.g. for a type of node $n$:

$$\bar{\mu_n} \leftarrow \mathbf{A}_k \mu\left(\mathbf{z}_k(t)\right),\ k \in \mathcal{N}_n \tag{8}$$

$$\mathbf{z}_n(t+1) = \phi(\mathbf{z}_n(t),\ \mu_n) \tag{9}$$

where (8) represents the accumulation of messages of all neighboring node types ($\mathcal{N}_n$), $\mu : \mathbb{R}^{d_k} \rightarrow \mathbb{R}^{d_n}$, where $d_x$ is the embedded dimension of a type $x$ node, $\mathbf{A}_k$ represents the adjacency matrix between type $n$ and type $k$ nodes, and $\mu_n$ represents the complete message accumulation of all neighboring nodes.

Considering that the electrical networks that we wish to analyze consist of three types of nodes, namely slack (S), generator (PV) and load (PQ) buses, the possibility to assign them to different types of nodes in the graph representation is advantageous; as well as being able to considering branches as a type of node in the graph representation (E), allowing to take the branch characteristics into account for the final model.

### B. TGN based Power Flow Solver

The proposed power flow solver is based on a specified number of TGN layers, each with three types of nodes: PV, PQ and branch nodes (E). Since the purpose of the NN is to infer the voltage phase and magnitude for each node, and this information is a priori available for the slack node, this node type is exclusively used between layers to update the input features of the following layer, i.e. the power balance error of the other node types depends, in part, on the voltage and generated power value of the slack node. The input features for each type of node are defined as follows:

$$\mathbf{x}_{PV} \begin{cases} V_{mag,i} \\ V_{ang,i} \\ \Delta P_i \end{cases} , \quad \mathbf{x}_{PQ} \begin{cases} V_{mag,j} \\ V_{ang,j} \\ \Delta P_j \\ \Delta Q_j \end{cases} , \quad \mathbf{x}_{E} \begin{cases} \gamma(R_k, X_k) \\ \delta(R_k, X_k) \\ B_k \end{cases} \tag{10}$$
$$\forall i \in \mathcal{V}_{PV}, \forall j \in \mathcal{V}_{PQ}, \forall k \in \mathcal{V}_{E}$$

where $\mathcal{V}_{PV}$, $\mathcal{V}_{PQ}$ and $\mathcal{V}_{E}$ correspond to generator, load and branch node type sets, respectively. $V_{mag,x}$ and $V_{ang,x}$ are the voltage magnitude and phase values of bus $x$, respectively. $\Delta P_x$ and $\Delta Q_x$ represent the active and reactive power equilibrium error of bus $x$, respectively. $R_k$, $X_k$ and $B_k$ represent the resistance, reactance and susceptance of edge $k$, respectively. The functions $\gamma$ and $\delta$ are defined by:

$$\gamma(R_k, X_k) = \frac{1}{\sqrt{R_k^2 + X_k^2}} \tag{11}$$

$$\delta(R_k, X_k) = \arctan(\frac{R_k}{X_k}) \tag{12}$$

The desired output features for each the PV and PQ nodes are defined as follows:

$$\mathbf{y}_{PV} \begin{cases} \hat{V}_{ang,i} \end{cases} , \quad \mathbf{y}_{PQ} \begin{cases} \hat{V}_{mag,j} \\ \hat{V}_{ang,j} \end{cases} \tag{13}$$
$$\forall i \in \mathcal{V}_{PV}, \forall j \in \mathcal{V}_{PQ}$$

where $\hat{V}_{mag,x}$ and $\hat{V}_{ang,x}$ are the inferred values of the voltage magnitude and phase of bus $x$, respectively.

Because of the way the TGN is defined, PV and PQ nodes cannot be directly connected with eachother, always having an E node (representing the branch) between them. Thus, only two adjacency matrices are defined:

$$\mathbf{A}_{PV \to E} \in \mathbb{R}^{|\mathcal{V}_{PV}| \times |\mathcal{V}_{E}|} \tag{14}$$

$$\mathbf{A}_{PQ \to E} \in \mathbb{R}^{|\mathcal{V}_{PQ}| \times |\mathcal{V}_{E}|} \tag{15}$$

where $|\cdot|$ represents the cardinality of a set. These matrices are transposed when the branch nodes are aggregating information from either generator or load buses (instead of the other way around).

A general schematic view of the proposed solver is shown in Fig. 1. As can be seen, the slack node is not directly represented as a graph node, but the $\Delta P$ and $\Delta Q$ values of the surrounding nodes are affected by the state of the slack node via the power equilibrium equations.

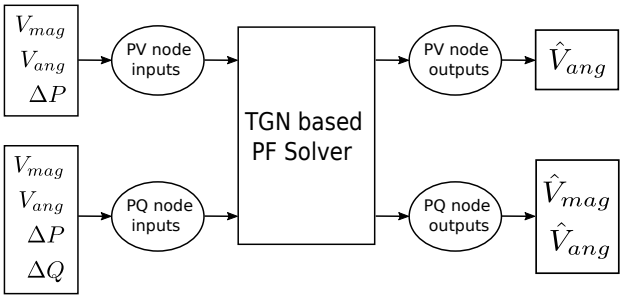


Fig. 1. Basic architecture of proposed solver.

As was mentioned previously, the entire TGN based solver is composed of a specified number of TGN layers, the outputs obtained at each level are analogous to a solution approximation obtained in an iteration of a numerical method such as Newton-Raphson. These outputs are used in the power equilibrium equations to obtain the TGN inputs of the next layer. This is illustrated in Fig. 2.

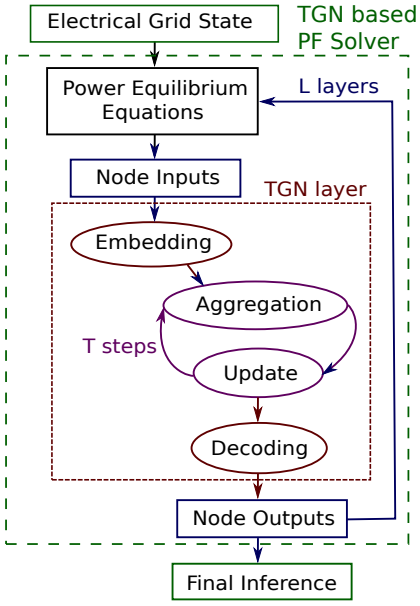


Fig. 2. Computational graph proposed solver.

### C. Training

The training of the parameters of all TGN layers is performed simultaneously and is based on minimizing the violation of the power equilibrium equations. This way, the loss function does not depend on target values obtained from other methods, but focuses on finding a solution that will minimize the active and reactive power imbalance at each

node. The training is performed in batches of $M$ samples; the loss function is defined as:

$$\frac{1}{M}\sum_{m=1}^{M}\left(\frac{1}{N}\sum_{n=1}^{N}\left(\Delta P_{n,m}^2+\Delta Q_{n,m}^2\right)\right) \tag{16}$$

where $N$ is the number of buses of the electrical grid.

Each sample is based on an IEEE power systems test case with the difference that a random branch in the test case is disconnected in each sample. This way the electrical grid configuration is different for each sample in the training batch, thus including the dynamic element to the power flow analysis.

## III. Test Results

In this section the procedure for testing the proposed solver is presented, including the way the input data was obtained and the hyperparameters of the system. To results obtained are compared with the results derived from Matpower with a Newton-Raphson based algorithm, since this is a conventional and trustworthy power flow solver.

The grid data was based on two standard IEEE power grids, which were then modified to vary the network configuration. The first grid tested is a modified version of the IEEE-30 bus system, with 40 branches (original case has 41, but random line outages were represented for each sample). The second is a modified version of the IEEE-118 bus system, with 79 branches (original case has 80). Since the focus of this work is on grid topology, and not so much on the fluctuation of the injections or line characteristics, not much variance was introduced for these values between the standard test cases and the generated data. For both test cases, uniform noise of only 5% of the original value was added to the injections, i.e. the active and reactive power demand, the active power generation and the generator voltage magnitude values. The same noise percentage was applied to the branch characteristics: resistance, reactance and susceptance.

Two instances of the TGN based power flow solver were trained, both with the same number of TGN layers, and aggregation and update steps. The models consist of 10 TGN layers ($L = 10$), this is a hyper-parameter chosen empirically to allow sufficient precision while keeping the complete model relatively small. The number of aggregation and update steps produces the exchange of information with the one-hop neighbors of each node, to avoid the distortion of information from distant nodes, only two aggregation and update steps were chosen ($T = 2$). On of the models is trained with data from the modified IEEE-30 node grid, and the other with data from the modified IEEE-118 node grid. The Adam optimizer with the default Tensorflow parameters was used, in batches of 20 samples ($M = 20$).

The proposed solver will always infer something no matter the state of the electrical grid, but because of the way the network samples are defined, some of the samples the model is trained on do not converge in Matpower with the Newton-Raphson based method; however, for testing, only samples that converged in Matpower are considered. Each instance of the proposed solver was tested on new grid samples with the same number of nodes as the ones it was trained on, and on grid samples of different size from the ones they were trained on. The results are summarized in tables I and II for the voltage magnitude and phase, respectively.

TABLE I
VOLTAGE MAGNITUDE MSE RESULTS

| **Voltage Magnitude MSE of Test Batch** | | |
|---|---|---|
| | ***Trained on IEEE-30*** | ***Trained on IEEE-118*** |
| **Tested on IEEE-30** | 9.7997e-05 | 1.8466e-03 |
| **Tested on IEEE-118** | 7.5182e-04 | 2.4996e-05 |

TABLE II
VOLTAGE PHASE MSE RESULTS

| **Voltage Phase MSE of Test Batch** | | |
|---|---|---|
| | ***Trained on IEEE-30*** | ***Trained on IEEE-118*** |
| **Tested on IEEE-30** | 1.1991e-03 | 1.5274e-03 |
| **Tested on IEEE-118** | 7.0574e-03 | 2.2512e-03 |

For illustrative purposes, Fig. 3 and Fig. 4 respectively show graphs of the voltage magnitude and phase comparison of a random sample of the solver that was trained and tested on modified versions of the IEEE-30 bus test case.

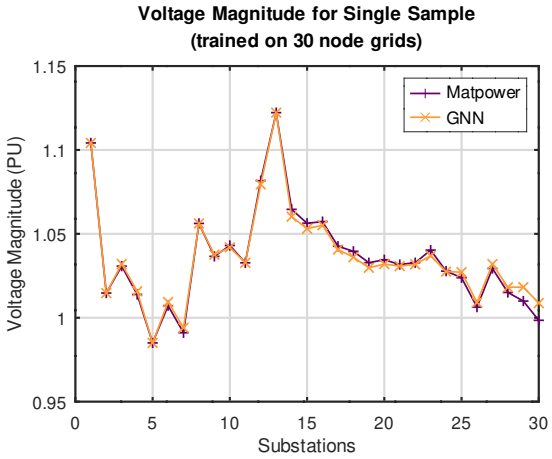


Fig. 3. Magnitude comparison, 30 node grid.

Fig. 5 and Fig. 6 respectively show graphs of the voltage magnitude and phase comparison of a random sample of the solver that was trained and tested on modified versions of the IEEE-118 bus test case.

## IV. Conclusion

The main contribution of this paper is the proposed power flow solver method based on a type of graph neural network that is able to exploit the differences between the types of nodes by considering independent multidimensional features for the different bus types present in an electrical power grid. A significant aspect of the presented work is the importance given to generalizing to different electrical grid topologies. To

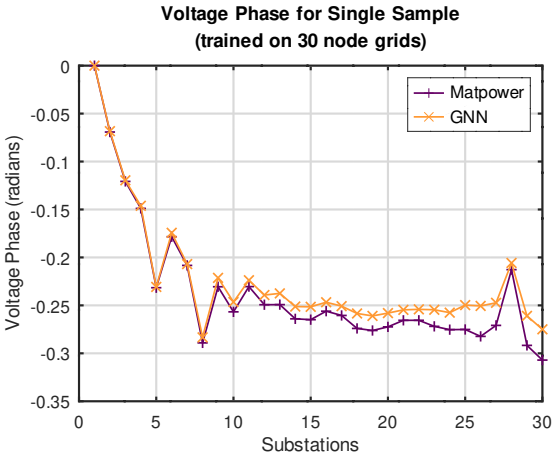


Fig. 4. Phase comparison, 30 node grid.

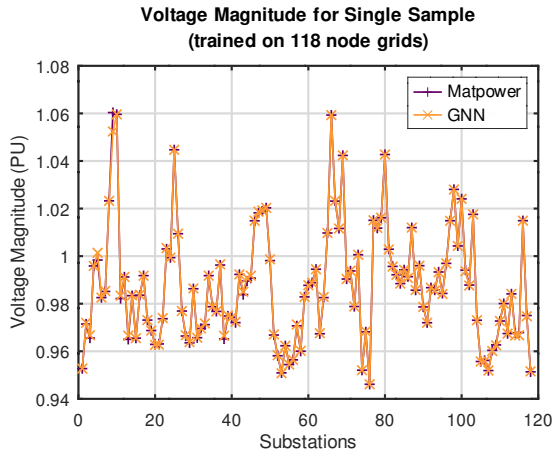


Fig. 5. Magnitude comparison, 118 node grid.

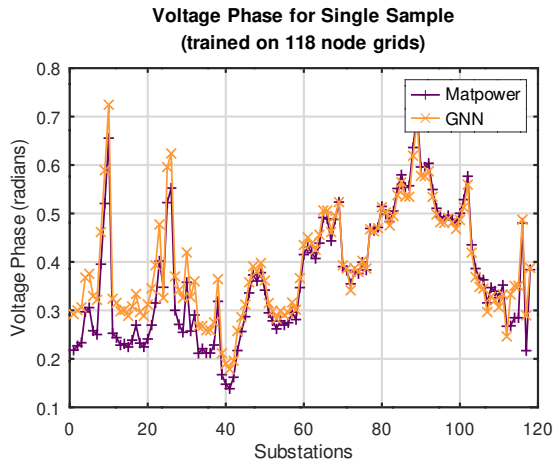


Fig. 6. Phase comparison, 118 node grid.

the best of the authors' knowledge, there is no other work that solves the power flow problem with neural networks employing different power grid configurations during the training of the parameters. The proposed method does not intend on imitating any other method, but is focused on minimizing the active and reactive power imbalance at each node of each sample during the training of the parameters.

The computation of Jacobian matrices and their inverse is completely avoided, as the method does not act directly on the voltage variables but rather learns the parameters of the update and aggregation functions of the proposed TGN model (as well as the encoding and decoding function parameters). For small grids the computational time is similar to Newton-Raphson based methods, however, GNNs scale linearly with the number of edges and embedding size, indicating that for larger grids, the presented method can lead to decreased computational time per power flow. The presented method is considerably more scalable and requires less memory storage than other FFNN based methods since it is based on local operations and shared modules, i.e. the size of the model does not depend on the size of the electrical grid.

By way of the numerical experiments, it was demonstrated that the proposed graph neural network based method obtains results very similar to the output of a conventional Newton-Raphson based method, even when these target values were in no way present during training.

The results presented constitute an important advance toward a neural network based system that can aid in improving the stages of power system planning, optimization, operation and control.

## REFERENCES


[1] L. Xie, C. Singh, S. K. Mitter, M. A. Dahleh, and S. S. Oren, "Toward carbon-neutral electricity and mobility: Is the grid infrastructure ready?" *Joule*, vol. 5, no. 8, pp. 1908–1913, 8 2021.

[2] J. D. Glover, M. S. Sarma, and T. J. Overbye, *Power System Analysis and Design*, 5th ed. Cengage Learning, 2012.

[3] M. Gholami, M. J. Sanjari, M. Safari, M. Akbari, and M. R. Kamali, "Static security assessment of power systems: A review," *International Transactions on Electrical Energy Systems*, vol. 30, no. 9, pp. 2050–7038, 2020.

[4] J. Grainger and W. Stevenson, *Power System Analysis*. McGraw-Hill, 1994.

[5] B. Stott, "Review of load-flow calculation methods," *Proceedings of the IEEE*, vol. 62, no. 7, pp. 916–929, 1974.

[6] A. J. Monticelli, A. Garcia, and O. R. Saavedra, "Fast decoupled load flow: hypothesis, derivations, and testing," *EEE Transactions on Power Systems*, vol. 5, no. 4, pp. 1425–1431, 11 1990.

[7] H. Saadat, *Power Systems Analysis*. McGraw-Hill, 2002.

[8] N. Amjady, "Dynamic voltage security assessment by a neural network based method," *Electric Power Systems Research*, vol. 66, no. 3, pp. 215–226, 9 2003.

[9] T. Jain, L. Srivastava, and S. N. Singh, "Fast Voltage Contingency Screening Using Radial Basis Function Neural Network," *IEEE Transactions on Power Systems*, vol. 18, no. 4, pp. 1359–1366, 11 2003.

[10] D. Thukaram, H. P. Khincha, and H. P. Vijaynarasimha, "Artificial neural network and support vector machine approach for locating faults in radial distribution systems," *IEEE Transactions on Power Delivery*, vol. 20, no. 2 I, pp. 710–721, 4 2005.

[11] L. Zhuang, H. Liu, J. Zhu, S. Wang, and Y. Song, "Comparison of forecasting methods for power system short-Term load forecasting based on neural networks," in *2016 IEEE International Conference on Information and Automation, IEEE ICIA 2016*. Institute of Electrical and Electronics Engineers Inc., 1 2017, pp. 114–119.

[12] G. W. Kim and K. Y. Lee, "Coordination contvol of ULTC transformer and STATCOM based on an artificial neural network," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 580–586, 5 2005.

[13] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," pp. 405–421, 4 2021.

[14] V. S. S. Vankayala and N. D. Rao, "Artificial neural networks and their applications to power systems—a bibliographical survey," *Electric Power Systems Research*, vol. 28, no. 1, pp. 67–79, 1993.

[15] N. Kumar, R. Wangneo, P. K. Kalra, and S. C. Srivastava, "Application Of Artificial Neural Networks To Load Flow Solutions," in *Region 10 International Conference on EC3-Energy, Computer, Communication and Control Systems*, 1991, pp. 199–203.

[16] X. Hu, H. Hu, S. Verma, and Z.-L. Zhang, "Physics-Guided Deep Neural Networks for Power Flow Analysis," *IEEE Transactions on Power Systems*, vol. 36, no. 3, pp. 2082–2092, 1 2021. [Online]. Available: http://arxiv.org/abs/2002.00097 http://dx.doi.org/10.1109/TPWRS.2020.3029557

[17] M. Fikri, B. Cheddadi, O. Sabri, T. Haidi, B. Abdelaziz, and M. Majdoub, "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., 10 2018.

[18] J. Skarding, B. Gabrys, and K. Musial, "Foundations and modelling of dynamic networks using Dynamic Graph Neural Networks: A survey," *IEEE Access*, vol. 9, pp. 79 143–79 168, 5 2021. [Online]. Available: http://arxiv.org/abs/2005.07496 http://dx.doi.org/10.1109/ACCESS.2021.3082932

[19] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural Message Passing for Quantum Chemistry," in *International Conference on Machine Learning*, JMLR.org, Ed., Sydney, 4 2017, pp. 1263–1272.

[20] M. O. R. Prates, P. H. C. Avelar, H. Lemos, M. Gori, and L. Lamb, "Typed Graph Networks," 1 2019. [Online]. Available: http://arxiv.org/abs/1901.07984