Distributed Power System State Estimation Using Graph Convolutional Neural Networks

SangWoo Park
UC Berkeley
spark111@berkeley.edu

Fernando Gama
Rice University
fgama@rice.edu

Javad Lavaei
UC Berkeley
lavaei@berkeley.edu

Somayeh Sojoudi
UC Berkeley
sojoudi@berkeley.edu

Abstract

State estimation plays a key role in guaranteeing the safe and reliable operation of power systems. This is a complex problem due to the noisy and unreliable nature of the measurements that are obtained from the power grid. Furthermore, the laws of physics introduce nonconvexity, which makes the use of efficient optimization-based techniques more challenging. In this paper, we propose to use graph convolutional neural networks (GCNNs) to learn *state estimators from data. The resulting estimators are distributed and computationally efficient, making them robust to cyber-attacks on the grid and capable of scaling to large networks. We showcase the promise of GCNNs in distributed state estimation of power systems in numerical experiments on IEEE test cases.*

1. Introduction

Power system state estimation aims to recover the system's current underlying voltage phasors, given supervisory control and data acquisition (SCADA) measurements, PMU measurements, and a system model based on assumed parameters [1]. State estimation not only helps prevent failures in the power network, but also underpins every aspect of real-time power system operation and control. The major challenges of state estimation arise from the fact that power system measurements are riddled with noise and corrupt data, and subject to missing values or cyber-attacks. Furthermore, due to the laws of physics, the measurement values are nonconvex functions of voltages, which limit the application of traditional convex optimization techniques that enjoy strong theoretical and performance guarantees. The most commonly implemented and well-studied method for state estimation is the weighted least squares (WLS) method, which is robust against Gaussian measurement noise. However, the performance of WLS deteriorates significantly in the presence of gross measurement errors or corrupt data. Moreover, efficient local-search algorithms such as Newton's method may not converge to the true state due to the existence of spurious local minima. In order to deal with gross errors, several different methods based on least absolute value minimization have been proposed [2–8]. To overcome nonconvexity, a convex relaxation based on semidefinite programming is proposed in [9] and [10]; and a method for finding a region around the global minimum that contains no spurious local minima is developed in [11].

Graph neural networks (GNNs) have emerged as effective learning architectures in distributed problems [12–14]. Consisting of a cascade of layers, each of which applies a graph filter followed by a nonlinear activation function, they have shown promising performance in a myriad of tasks involving robotics, distributed control, sensor networks and communications. The success of GNNs in these tasks can be traced back to their distributed nature and their ability to exploit the underlying data structure. More concretely, they have exhibited strong scalability properties as well as robustness to changes in the underlying graph support [15].

In this paper, we propose to use a specific type of GNN, namely Graph Convolutional Neural Network (GCNN), for power system state estimation based on SCADA and PMU measurements. GCNNs leverage the underlying graph structure of the power grid to compute the estimates by exchanging information among buses that share a line. This leads to a distributed architecture that will be shown to be robust to cyber-attacks in the power grid. In addition, GCNNs are computationally efficient, resulting in estimators that scale up to larger power networks. GCNNs can be trained in a supervised manner by minimizing the mean square loss, and then are shown to generalize well to unseen measurement scenarios.

In this paper, we introduce a comprehensive framework for utilizing learning architectures in the problem of power system state estimation. This was first done in [16] by using an EdgeNet [17]. EdgeNets have the drawback of requiring too many parameters (scaling with the size of power network), and thus demanding very large training sets. Furthermore, in [16] only one-hop information is considered on each layer. We propose to use GCNNs based on finite impulse-response (FIR) graph filters (graph convolutions) that scale better and train faster. By using repeated communications with one-hop neighbors, the proposed GCNN method is also capable of including information from further away neighbors, leading to better estimates. The contributions of this paper can be summarized as follows:

- We use GCNNs to learn power system state estimators from SCADA measurement and PMU measurements.
- We discuss how the properties of the GCNN make them sensible learning architectures for this problem. In particular, we focus on the distributed nature of the architecture, its computational efficiency and its scalability.
- Through extensive numerical experiments, we showcase the satisfactory performance of state estimators learned from GCNNs.
- We also show how the distributed nature of GCNNs makes the resulting state estimators robust to cyber-attacks.

2. State Estimation

Consider a power grid described by a graph $\mathcal{G} = \{\mathcal{B}, \mathcal{L}\}$ where the set of nodes $\mathcal{B} = \{b_1, \ldots, b_N\}$ corresponds to N buses in the grid and the edge set $\mathcal{L} = \{c_1, \ldots, c_M\} \subseteq \mathcal{B} \times \mathcal{B}$ corresponds to the lines connecting the buses, i.e. $(b_i, b_j) \in \mathcal{L}$ if and only if there is a line connecting bus b_i with bus b_j. The set of buses connected to each bus b_i is denoted by the neighborhood set $\mathcal{N}_i = \{b_j \in \mathcal{B} : (b_j, b_i) \in \mathcal{L}\}$. It is noted that, given the bidirectional nature of power lines, the resulting graph $\mathcal{G}$ is undirected, i.e. $(b_i, b_j) \in \mathcal{L} \Leftrightarrow (b_j, b_i) \in \mathcal{L}$. The connectivity of the network can also be captured by the incidence matrix $\mathbf{B} \in \mathbb{R}^{N \times M}$, defined as

$$\mathbf{B}_{ij} = \begin{cases} -1 & \text{if } c_j = (i,k) \text{ for some } k \in \mathcal{B} \\ 1 & \text{if } c_j = (k,i) \text{ for some } k \in \mathcal{B} \\ 0 & \text{otherwise} \end{cases}$$

The state of a power system is captured by the complex-valued voltages at the network buses, which can be characterized by their voltage magnitudes and voltage angles at any given time. Denote by $v_i \in \mathbb{C}$ the complex voltage at bus $b_i \in \mathcal{B}$, with magnitude $|v_i| \in \mathbb{R}_+$ and angle $\measuredangle v_i \in [-\pi, \pi)$. Analogously, denote by $s_i \in \mathbb{C}$ the complex power injection into node $b_i \in \mathcal{B}$, with the active power denoted as $p_i \in \mathbb{R}$ and the reactive power denoted by $q_i \in \mathbb{R}$, such that $s_i = p_i + \mathbf{j} q_i$, where $\mathbf{j}$ is the imaginary unit. Let $\mathbf{v}, \mathbf{s} \in \mathbb{C}^N$ and $\mathbf{p}, \mathbf{q} \in \mathbb{R}^N$ denote the vectors of nodal voltages, complex powers, active powers and reactive powers, respectively. In a slight abuse of notation, the vectors $|\mathbf{v}| \in \mathbb{R}^N$ and $\measuredangle \mathbf{v} \in \mathbb{R}^N$ are defined in such a way that $[|\mathbf{v}|]_i = |v_i|$ and $[\measuredangle \mathbf{v}]_i = \measuredangle v_i$.

The physical characteristics of the grid are captured by an admittance matrix. More specifically, each bus b_i is accompanied by a nodal shunt admittance $\hat{y}_i = \hat{y}_i^{\mathfrak{Re}} + \mathbf{j}\hat{y}_i^{\mathfrak{Im}}$, while each line (b_i, b_j) is represented by a Π-model with a series admittance $\tilde{y}_{ij} = \tilde{y}_{ij}^{\mathfrak{Re}} + \mathbf{j}\tilde{y}_{ij}^{\mathfrak{Im}}$ and the total shunt admittance $\check{y}_{ij} = \check{y}_{ij}^{\mathfrak{Re}} + \mathbf{j}\check{y}_{ij}^{\mathfrak{Im}}$. These values can be conveniently captured in the admittance matrix $\mathbf{Y} \in \mathbb{C}^{N \times N}$ defined as

$$[\mathbf{Y}]_{ij} = \begin{cases} \hat{y}_i + \sum\limits_{k: b_k \in \mathcal{N}_i} (\tilde{y}_{ik} + \check{y}_{ik}/2) & \text{if } i = j, \\ -\tilde{y}_{ij} & \text{if } (i,j) \in \mathcal{L}, \\ 0 & \text{otherwise.} \end{cases}$$

The notation $[\mathbf{M}]_{ij}$ denotes the (i,j) entry of a matrix $\mathbf{M}$. Note that the admittance matrix can be decomposed into its real and imaginary parts $\mathbf{Y} = \mathbf{Y}^{\mathfrak{Re}} + \mathbf{j}\mathbf{Y}^{\mathfrak{Im}}$ for $\mathbf{Y}^{\mathfrak{Re}} \in \mathbb{R}^{N \times N}$ and $\mathbf{Y}^{\mathfrak{Im}} \in \mathbb{R}^{N \times N}$.

The voltage and powers at each bus are related via the power flow equations. More specifically, for each bus $b_i \in \mathcal{B}$, we have

$$p_i = \sum_{j=1}^{N} |v_i||v_j| \big(y_{ij}^{\mathfrak{Re}} \cos \measuredangle v_{ij} + y_{ij}^{\mathfrak{Im}} \sin \measuredangle v_{ij} \big) \quad (1a)$$

$$q_i = \sum_{j=1}^{N} |v_i||v_j| \big(y_{ij}^{\mathfrak{Re}} \sin \measuredangle v_{ij} - y_{ij}^{\mathfrak{Im}} \cos \measuredangle v_{ij} \big) \quad (1b)$$

where $y_{ij}^{\mathfrak{Re}} = [\mathbf{Y}^{\mathfrak{Re}}]_{ij}$ and $y_{ij}^{\mathfrak{Im}} = [\mathbf{Y}^{\mathfrak{Im}}]_{ij}$ are the real and imaginary parts of the admittance matrix, respectively.

The problem of state estimation in power grids consists in obtaining a complete characterization of the bus voltages $\mathbf{v} \in \mathbb{C}^N$ from noisy and tainted measurements given a known system model. To streamline the presentation and without loss of generality, we begin by assuming that the measurements are obtained from the SCADA system, and include voltage magnitude $\widehat{|\mathbf{v}|}$, active power $\widehat{\mathbf{p}}$ and reactive power $\widehat{\mathbf{q}}$ at all buses, where the notation $\widehat{\cdot}$ is used to indicate the potential inaccuracy in the measurement values (see Section 4.5 for incomplete measurements, PMU measurements as well as the case with line measurements). In other words, the method to be developed can handle any arbitrary measurement set but to simplify the presentation we explain the ideas for a canonical measurement set. Putting all available measurements into a vector $\hat{\mathbf{x}}$, one can write

$$\hat{\mathbf{x}} = \mathsf{m}_x(\mathbf{v}) + \boldsymbol{\eta} + \boldsymbol{\omega} \tag{2}$$

where $\mathsf{m}_x : \mathbb{C}^N \to \mathbb{R}^{3N}$ is the measurement function, which is based on the power flow equation (1) for power measurements as well as an operator that outputs the magnitude of a vector $\mathbf{v}$ for voltage magnitude measurements. The term $\boldsymbol{\eta} \in \mathbb{R}^{3N}$ models the random measurement noise and $\boldsymbol{\omega} \in \mathbb{R}^{3N}$ models the sparse gross errors (corrupt data) that can be potentially caused by cyber-attacks. An estimator is then defined as a function $\Phi : \mathbb{R}^N \times \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^N \times \mathbb{R}^N$ that takes the measurements $\widehat{|\mathbf{v}|}$, $\widehat{\mathbf{p}}$ and $\widehat{\mathbf{q}}$ and returns the estimates of the voltage magnitudes $|\mathbf{v}|$ and voltage angles $\measuredangle\mathbf{v}$ at all buses. The performance of the estimator is evaluated by the mean squared error (MSE).

In what follows, GCNNs are leveraged to parametrize the estimator Φ. Available data can then be used to train the GCNN (i.e. choose the appropriate parameter values).

3. Graph Neural Networks

Given a graph $\mathcal{G} = \{\mathcal{B}, \mathcal{L}\}$, a graph signal with F features can be conveniently described by a matrix $\mathbf{X} \in \mathbb{R}^{N\times F}$ where the i^{th} row corresponds to the feature values at the i^{th} node $\mathbf{x}_i \in \mathbb{R}^F$. Each column $\mathbf{x}^f \in \mathbb{R}^N$ collects the f^{th} feature entry of the signal across all nodes.

Describing the graph signal as a matrix $\mathbf{X} \in \mathbb{R}^{N\times F}$ is mathematically convenient, but loses track of the connection between the signal values and the underlying graph support, i.e. there is no information about the graph in the matrix $\mathbf{X}$. To recover this information, a graph matrix description (GMD) $\mathbf{S} \in \mathbb{R}^{N\times N}$ is used [18]. This matrix must share the sparsity pattern of the underlying graph, i.e. $[\mathbf{S}]_{ij} = 0$ if $(b_j, b_i) \notin \mathcal{L}$ for $i \neq j$. Common choices for the GMD include the Laplacian, Markov and adjacency matrices of the graph. The GMD can be used to relate the graph signal $\mathbf{X}$ to the underlying graph support, by defining the most basic linear operation between graph signals, $\mathsf{S} : \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times F}$ [18]. Namely, $\mathbf{Z} = \mathsf{S}(\mathbf{X}) = \mathbf{S}\mathbf{X}$ so that for each (i, f) element of $\mathbf{Z}$, we have

$$[\mathbf{Z}]_{if} = \sum_{j: b_j \in \mathcal{N}_i \cup \{b_i\}} [\mathbf{S}]_{ij} [\mathbf{X}]_{jf}. \tag{3}$$

The equality is given by the definition of matrix multiplication. Note, however, that due to the sparsity pattern of $\mathbf{S}$, the summation is only over the buses that share a connection with bus b_i. This fact implies that pre-multiplying a graph signal $\mathbf{X}$ by $\mathbf{S}$ leads to a distributed operation, where each node can compute its output by relying only on the information shared by their one-hop neighbors. Thus, $\mathbf{SX}$ compactly represents the linear, distributed operation of sharing information between neighbors and linearly combining it.

The operation in (3) is the basic block for building linear graph filters. In particular, a finite impulse response (FIR) graph filter $\mathsf{H} : \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times G}$ is defined as:

$$\mathsf{H}(\mathbf{X}; \mathbf{S}, \mathcal{H}) = \sum_{k=0}^{K} \mathbf{S}^k \mathbf{X} \mathbf{H}_k \tag{4}$$

where $\mathbf{H}_k \in \mathbb{R}^{F\times G}$ are known as the filter coefficients or filter taps. The set $\mathcal{H} = \{\mathbf{H}_k \mid k = 0, \ldots, K\}$ contains the $(K+1)$ filter taps that, together with the GMD $\mathbf{S}$, completely characterize the graph filter. This filter is also known as a graph convolution due to its sum-and-shift nature [12].

The graph filter in (4) is also a distributed operation. Note that the pre-multiplication of $\mathbf{X}$ by $\mathbf{S}^k$ actually entails k repeated exchanges with one-hop neighbors. The post-multiplication by $\mathbf{H}_k$, on the other hand, only carries out a linear combination of the rows of $\mathbf{X}$ which correspond to information contained by each node and does not entail any exchanges between nodes. It is worth emphasizing that equation (4) represents a compact mathematical notation for describing the sharing of information (and posterior linear combination) among nodes, but it does not mean that the powers of $\mathbf{S}$ are required to be able to compute the output. Furthermore, the nodes do not even require complete knowledge of $\mathbf{S}$, since they only need to exchange information with their immediate neighbors.

The graph filter in (4) is linear in the input $\mathbf{X}$ and, therefore, it is only able to capture linear relationships between input and output. Nonlinear relationships can be captured by means of a GCNN $\Phi : \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times G}$, which is defined as a cascade of L layers (blocks), each of which applies a graph filter (4) followed by a nonlinear and pointwise activation function

$$\Phi(\mathbf{X}; \mathbf{S}, \mathcal{H}) = \mathbf{X}_L, \;\; \mathbf{X}_\ell = \sigma\big(\mathsf{H}_\ell(\mathbf{X}_{\ell-1}; \mathbf{S}, \mathcal{H}_\ell)\big) \tag{5}$$

for $\ell = 1, \ldots, L$ and where, in an abuse of notation, $[\sigma(\mathbf{Z})]_{ij} = \sigma([\mathbf{Z}]_{ij})$ denotes the entrywise application of a scalar activation function $\sigma : \mathbb{R} \to \mathbb{R}$. Note that $\mathbf{X}_0 = \mathbf{X}$ is the input and $\mathbf{X}_L$ is the output, which entails that $F_0 = F$ and $F_L = G$. In (5), the set $\mathcal{H} = \{\mathcal{H}_\ell \mid \ell = 1, \ldots, L\}$ contains the filter taps of all layers and, given the GMD $\mathbf{S}$, it completely characterizes the representation space of the GCNN.

The GCNN (5) inherits the distributed nature from graph filters, since the point-wise activation functions do not involve any operation on the GMD. GCNNs of this form also exhibit the properties of permutation invariance, Lipschitz continuity to changes in the GMD $\mathbf{S}$ [19], and enhanced processing capabilities with respect to linear graph filters [15]. These properties play a key role in their ability to transfer to new scenarios and scale up to larger graphs [12].

The number of layers L, the order K_ℓ of the filters and the number of features F_ℓ at each layer, as well as the pointwise activation function σ, are all design choices and are commonly known as hyperparameters. While several methods for selecting hyperparameters are available, they largely remain a choice of the practitioner. The specific filter taps $\mathcal{H}$, on the other hand, are usually learned from available data by a procedure known as training. During training, a proxy optimization problem is solved to find the set of parameters (filter taps) that minimize some loss function. Typically, this optimization problem is solved by means of some algorithm based on stochastic gradient descent (SGD), thus requiring that the loss function be differentiable. Note that the total number of parameters is $\sum_{\ell=1}^{L} K_\ell F_\ell F_{\ell-1}$, independent of the size of the graph N.

Unlike traditional optimization problems, the goal is not necessarily to find the minimum of the training optimization problem, but rather to use it as a guide in finding suitable parameters that will perform adequately on new data, unobserved during training (generalization) [20, Sec. 8.1]. Therefore, the performance evaluation of the trained algorithm is not determined by how low the value of the loss function is, but rather how good the algorithm is at solving a task (in this case, state estimation) on data that was not observed during training. This implies that the training of the algorithm is carried out in a centralized manner (which is actually required, due to parameter sharing), without necessary violating the distributed nature of the algorithm at evaluation time.

4. GCNN for Power System State Estimation

In this section, we will first analyze a simplified version of state estimation that utilizes only nodal measurements from the grid, and assumes that all of the data are available. We propose a GCNN for this purpose (called GCNN-nodal), explain the training process, and study the performance of the trained neural network in the presence of noise and cyber-attacks. Then, we extend this framework to the full state estimation that utilizes nodal, line and PMU measurements. For the full state estimation, we examine two different GCNNs that use different types of GMD for the line data (GCNN-hodgeLap and GCNN-edgeLap).

4.1. GCNN for Nodal Measurements without Missing values

The neural network that we design consists of three graph convolutional layers and one simple linear filter. The first graph signal, essentially the input to the neural network $\mathbf{X}_0 \in \mathbb{R}^{N\times 3}$, consists of three features ($F_0 = 3$), one for each type of nodal measurements. The number of features for the output signal is two ($F_4 = 2$) to account for the estimate of voltage magnitudes and voltage angles. The number of features for the other graph signals in between the layers is a design choice. The specific value of these numbers will be stated later in the paper.

As mentioned earlier, common choices for the GMD $\mathbf{S}$ include the Laplacian, Markov and adjacency matrices of the underlying graph. These matrices are able to capture the connectivity of the network but fail to capture the admittance values of the lines which directly affect the power system measurements. In the earlier work [21], the authors used the Gaussian kernel of the admittance matrix as the GMD. In this paper, instead of having a single GMD, we propose using two GMDs: the real and imaginary parts of $\mathbf{Y}$.

$$\mathbf{S}_1 = \mathfrak{Re}(\mathbf{Y}), \quad \mathbf{S}_2 = \mathfrak{Im}(\mathbf{Y}) \tag{6}$$

By doing this, we eliminate the ambiguity caused by the Gaussian kernel, i.e., two admittance matrices with different real and imaginary parts can still have the same Gaussian kernel.

Now, the graph convolutional layers of our proposed GCNN will be updated from the form in (5) to the following form which incorporates multiple GMDs or multiple edge features

$$\mathbf{X}_\ell = \sigma\left(\sum_{e=1}^{2}\sum_{k=0}^{K} \mathbf{S}_e^k \mathbf{X}_{\ell-1} \mathbf{H}_{\ell k}^e\right) \quad \ell = 1, 2, 3. \tag{7}$$

Here, the choice of the nonlinear activation function is the hyperbolic tangent. Also, the superscript k on the GMD denotes the matrix power, whereas the superscript e on the filter matrix denotes the additional index accounting for multiple GMDs [17]. Finally, $\mathbf{X}_3$

is passed through a simple linear filter $\mathbf{H}_4 \in \mathbb{R}^{F_3 \times 2}$ in order to output the state estimate $\mathbf{X}^* \in \mathbb{R}^{N \times 2}$:

$$\mathbf{X}^* = \mathbf{X}_3 \mathbf{H}_4 \tag{8}$$

4.2. Training of GCNN

We construct datasets based on the IEEE power system test cases. Each dataset consists of multiple samples of randomly generated voltage magnitudes and angles (accounting for the true underlying state), and nodal measurements at all buses. Voltage magnitudes are sampled from a normal distribution with mean 1 and standard deviation 0.02, whereas voltage angles (in degrees) are sampled from a normal distribution with mean 0 and standard deviation 10. The measurement values are computed by using the sampled voltages as inputs to equation (1), which is then added with Gaussian noise. This leads to a labeled dataset, of which 81% is used for training, 9% for validation and the rest for testing.

During training, the objective is to minimize the MSE loss function between the estimate and the true voltages in the training set. The filter taps of the GCNN are updated by taking steps in the gradient descent direction using ADAM, an adaptive learning rate optimization algorithm for training deep neural networks [22]. The gradient is estimated using batches of 20 samples, and the whole training procedure is repeated for 40 epochs. Every 5 training steps (parameter updates), a validation stage is run, where the evaluation measure is computed over the validation set. The results of the validation stage in Fig. 1 show that the GCNN-based estimators are indeed learning, improving their performance.

During testing, the performance of the trained GCNN is measured by the RMSE, which is defined as the following

$$\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \frac{1}{\sqrt{N}} \|\mathbf{X}^*_{rec} - \mathbf{X}^{true}_{rec}\| \tag{9}$$

where $\mathbf{X}^*_{rec}$ is the estimated state in rectangular coordinates, $\mathbf{X}^{true}_{rec}$ is the true underlying state in rectangular coordinates and $\|\cdot\|$ denotes the ℓ_2 norm of the vectorized argument.

4.3. Performance of GCNN for Noisy Measurements

The performance of a state estimator can be assessed by computing the root mean squared error (RMSE) on the test dataset. In order to evaluate the efficacy of GCNNs, we compare its performance to three other methods. The first method is the nonlinear weighted least squares (WLS) method where the nominal point in power systems is used as an initial point of the algorithm. The second method is the GraphPrune method proposed in [16]. We design the number of neurons in each layer so that the total number of parameters matches that of the GCNN. Finally, we also use the state estimate obtained from GCNN as an initial point of the WLS method. This can be viewed as combining neural networks with optimization. Table 1 summarizes the result for the numerical experiments. We observe that the proposed GCNN method outperforms the GraphPrune method and also provides a good initial point for the WLS method. For the IEEE 39 system, the GCNN even outperforms the WLS because in this case, the nominal point turns out to be a bad initial point for a handful of states that we wish to recover. For networks larger than the 57 bus system, an effective neural network could not be trained for the GraphPrune method using our computational capacity of 12GB GPU. The observation is that our method works as well as the traditional WLS method in a normal scenario without outliers in the data and that its training can be done much faster than GraphPrune. One main benefit of our method over the over methods will be demonstrated next in the case when a subset of the data is grossly wrong.

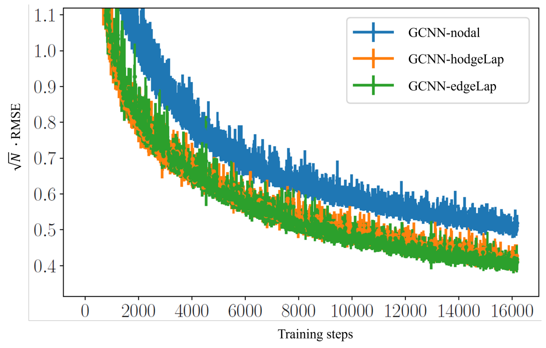
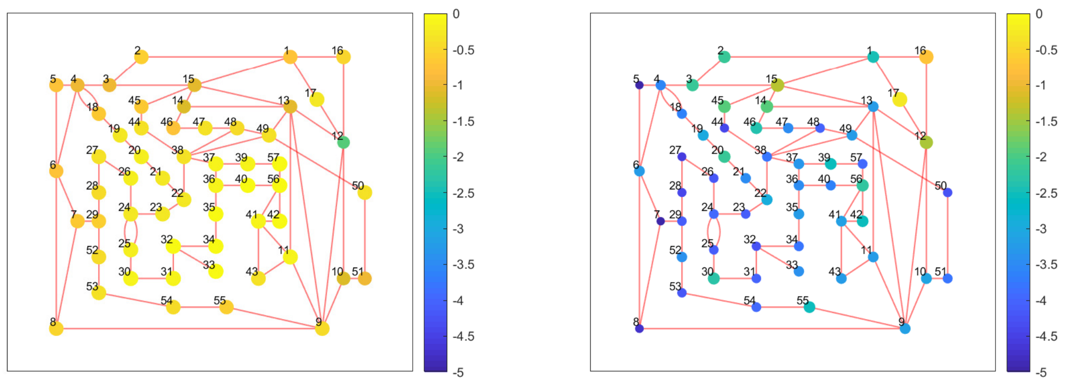


Figure 1: RMSE values in the validation stage. It is observed that with the progression of training steps (parameter updates through ADAM), the RMSE lowers, showing that the GCNNs are indeed learning.

Table 1: The value of RMSE averaged over five data splits. The GCNNs were trained using a dataset of 10,000 number of samples. The following parameter values were used for GCNN: $K = 2, F_1 = 100, F_2 = 400, F_3 = 200$. The following parameter values were used for GraphPrune: $F_1 = 20, F_2 = 10$.

Power network	WLS	GCNN	GCNN+WLS	GraphPrune
IEEE 39	0.3410	0.0304	0.0229	0.0799
IEEE 57	0.0243	0.0393	0.0238	0.0692
IEEE 118	0.0746	0.0293	0.0552	N/A



(a) WLS method **(b)** GCNN method

Figure 2: The two plots above summarize the state estimation errors for the WLS method and the GCNN method on the IEEE 57 bus system. In this cyber-attack scenario, the measurement values on four buses (#2, #15, #16, #17) are distorted from the actual values. The colors represent the log10 applied value of the state estimation error.

Table 2: The average absolute value of filter coefficients. This particular GCNN was trained on the IEEE 39 bus system using a datatset of 15,000 number of samples with the following parameter values: $K = 2, F_1 = 128, F_2 = 512, F_3 = 256$.

Average filter coeff. magnitudes	$\ell = 1$	$\ell = 2$	$\ell = 3$
$k = 0$	0.155	0.037	0.018
$k = 1$	1.100	0.040	0.026
$k = 2$	0.658	0.036	0.025

4.4. Performance of GCNN under Cyber-attacks

In addition to the high-quality state estimation under modest values of noise, we observe that the proposed GCNN is resilient against cyber-attacks that intentionally distort the measurement values in order to sabotage the system. To be more specific, even if a subset of the measurements is attacked, the negative effect can be isolated in a small neighborhood of the attacked region, which is not true for the other methods mentioned above. From numerical experiments, we observed that GraphPrune achieves some degree of defense against cyber-attacks but is far less effective than that of GCNN and has a high training complexity. In Figure 2, we simulate a cyber-attack scenario by distorting the measurement values at four buses in the IEEE 57 bus network. The color of a node in the graph represents the log10 value of the state estimation error at that node. For the WLS method, we can observe that the effect of the attack on four buses perpetrate throughout the entire network and can potentially derail the system. However, under the GCNN method, the effect of the attack does not spread throughout the network and is isolated around the few nodes that were attacked. This is due to the distributed nature of GCNNs as discussed in the previous sections. Furthermore, the GCNN is able to learn that the signals coming from buses that are closer to a certain bus have a stronger influence on that bus than those coming from buses that are further away. This result can be explained by the magnitudes of the filter taps, as summarized in Table 2.

4.5. Extension to PMU and Line Measurements

The GCNN architecture that we introduced represents a canonical form that uses only nodal SCADA measurements. For real power system data, there are also PMU and line power flow measurements. PMUs are useful devices that can directly measure the voltage magnitude and phase at a given time. However, due to their expensive costs, there is still a limited number of PMUs installed across the grid. For each node with a PMU measurement, the deviation of the estimated state from the PMU measurement can be added as a penalty in the loss function.

Power flows along lines are related to voltages via the power flow equations below, where p_{ij} and q_{ij} denote the active and reactive power flow from node i to j, respectively.[1] For each line, the power flow can be measured from either side (from-node and to-node) of the line, and therefore there are a total of four types of

[1]It is assumed that the shunt admittance values are zero

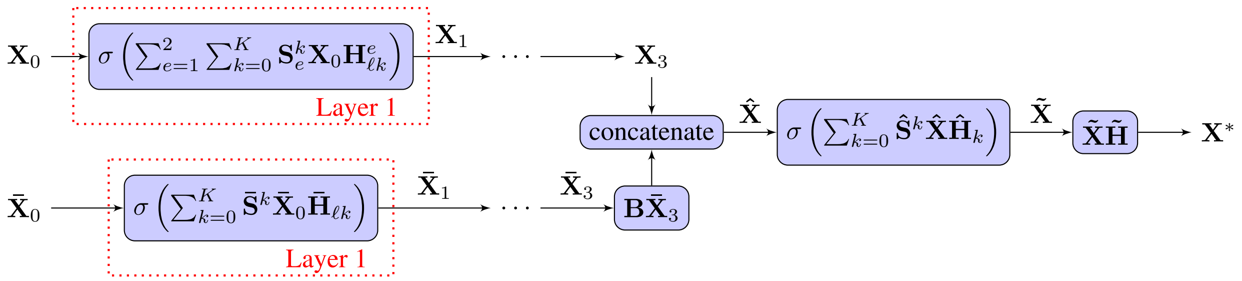

$\mathbf{X}_0$
$\sigma\left(\sum_{e=1}^{2}\sum_{k=0}^{K}\mathbf{S}_e^k\mathbf{X}_0\mathbf{H}_{\ell k}^e\right)$
Layer 1
$\mathbf{X}_1$
$\mathbf{X}_3$
concatenate
$\hat{\mathbf{X}}$
$\sigma\left(\sum_{k=0}^{K}\hat{\mathbf{S}}^k\hat{\mathbf{X}}\hat{\mathbf{H}}_k\right)$
$\tilde{\mathbf{X}}$
$\tilde{\mathbf{X}}\tilde{\mathbf{H}}$
$\mathbf{X}^*$
$\bar{\mathbf{X}}_0$
$\sigma\left(\sum_{k=0}^{K}\bar{\mathbf{S}}^k\bar{\mathbf{X}}_0\bar{\mathbf{H}}_{\ell k}\right)$
Layer 1
$\bar{\mathbf{X}}_1$
$\bar{\mathbf{X}}_3$
$\mathbf{B}\bar{\mathbf{X}}_3$

Figure 3: Flow-chart of the GCNN architecture with both node and line measurements. The dots in the chart account for the multiple layers (layers 2 and 3) that exist before the concatentation step.

line measurements.

$$p_{ij} = -y_{ij}^{\mathfrak{Re}}(|v_i|^2 - |v_i||v_j|\cos(\measuredangle v_{ij})) + y_{ij}^{\mathfrak{Im}}|v_i||v_j|\sin(\measuredangle v_{ij}) \quad (10a)$$

$$q_{ij} = y_{ij}^{\mathfrak{Im}}(|v_i|^2 - |v_i||v_j|\cos(\measuredangle v_{ij})) + y_{ij}^{real}|v_i||v_j|\sin(\measuredangle v_{ij}) \quad (10b)$$

Line power flow measurements are associated with graph edges and therefore they cannot be incorporated as nodal features. This issue can be circumvented by introducing additional nodes to the graph whenever we have line measurements. To illustrate this better, consider a line measurement for an active power flowing from node i to node j, namely p_{ij}. We introduce two new nodes between i and j, named auxiliary nodes k and m. We connect i to k, k to m, m to j, and then delete the edge (i, j). The three new lines have the same impedance values as the original line (i, j). Let the voltage at node k be equal to v_j and the voltage at node m be equal to v_i. Then, the line measurement p_{ij} corresponds to the nodal measurement at the new node k divided by 2. In other words, line features can be transformed into nodal features for an expanded graph. Although this provides a viable approach to include line measurements, it clearly increases the network size and therefore also increases the computational complexity of the training and estimating process. Furthermore, the topology of the network changes with the specific set of line measurements at hand, which can make the approach difficult to implement in practice.

Instead we train a GCNN for line signals, in parallel to that of node signals, that will be combined with node signals in the final step. Similar to the graph signal $\mathbf{X}$ defined on nodes, describe the graph signal on lines as a matrix $\bar{\mathbf{X}} \in \mathbb{R}^{M\times\bar{F}}$. The GMD accounting for line interactions $\bar{\mathbf{S}} \in \mathbb{R}^{M\times M}$ must share the sparsity pattern of the underlying graph in terms of the edge space. In this case, $[\bar{\mathbf{S}}]_{ij} = 0$ if lines c_i and c_j do not share a node. As indicated in [23], an enticing choice of GMD would be the *linegraph Laplacian*, constructed by treating each edge in the original graph as a node in the new graph and treating the incidence between edges in the original graph as an edge in the new graph. We denote this Laplacian matrix by $\mathbf{L}_{LG}$. Another potential choice of GMD is the *Hodge Laplacian*, which can be described in terms of the graph *incidence matrix* $\mathbf{B}$:

$$\mathbf{L}_{HG} = \mathbf{B}^T\mathbf{B} \quad (11)$$

Note that $\mathbf{L}_{LG}$ and $\mathbf{L}_{HG}$ share the same sparsity pattern.

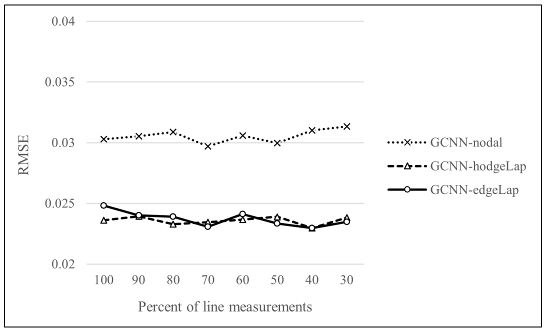
The neural network for handling line measurements again consists of three graph convolutional layers. The first graph signal, essentially the input to the neural network $\bar{\mathbf{X}}_0 \in \mathbb{R}^{M\times 4}$, consists of four features ($\bar{F}_0 = 4$), one for each type of line measurements. The number of features for the second and third layers are design choices. Now, the graph convolutional layers of our proposed GCNN will be of the form in (5)

$$\bar{\mathbf{X}}_\ell = \sigma\left(\sum_{k=0}^{\bar{K}}\bar{\mathbf{S}}^k\bar{\mathbf{X}}_{\ell-1}\bar{\mathbf{H}}_{\ell k}\right) \quad \ell = 1, 2, 3 \quad (12)$$

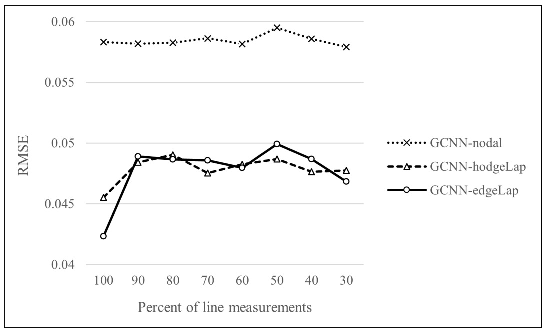
where $\bar{\mathbf{X}}_\ell \in \mathbb{R}^{M\times\bar{F}_\ell}$ denotes the edge graph signals with $\bar{F}_\ell$ features and $\bar{\mathbf{H}}_{\ell k} \in \mathbb{R}^{\bar{F}_{\ell-1}\times\bar{F}_\ell}$ denotes the filter coefficients. As before, the superscript k on the GMD, $\bar{\mathbf{S}}$, denotes the matrix power. Next, in order to convert edge signals to node signals, we multiply $\bar{\mathbf{X}}_3$ by the incidence matrix $\mathbf{B}$. Finally, the outputs of the graph convolutional layers for nodes (i.e., $\mathbf{X}_3$) and edges (i.e., $\mathbf{B}\bar{\mathbf{X}}_3$) are concatenated along columns and passed through an additional graph convolutional layer to produce $\tilde{\mathbf{X}}$, which is then passed through a simple linear filter $\tilde{\mathbf{H}} \in \mathbb{R}^{(F_3+\bar{F}_3)\times 2}$ in order to output the state estimate $\mathbf{X}^* \in \mathbb{R}^{N\times 2}$.

$$\mathbf{X}^* = \tilde{\mathbf{X}}\tilde{\mathbf{H}} \quad (13)$$

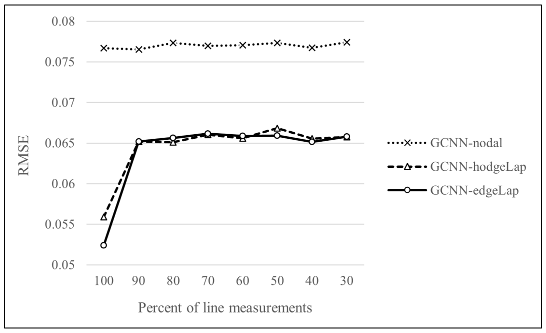
The process is summarized as a flow-chart in Figure 3.



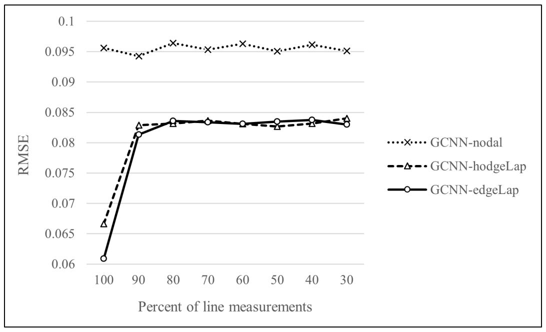
(a) 100% nodal measurements



(b) 90% nodal measurements



(c) 80% nodal measurements



(d) 70% nodal measurements

Figure 4: This figure compares the performance of state estimation for the IEEE 39 bus system between three methods: GCNN with only nodal measurements (GCNN-nodal), GCNN with line measurements using the hodge-Laplacian matrix (GCNN-hodgeLap) and GCNN with line measurements using the linegraph-Laplacian (GCNN-edgeLap). Each subfigure plots the RMSE value for a different percentage level (ranging from 100% to 70%) of available nodal measurements. Each point in the subfigure denotes the value of RMSE averaged over five data splits (i.e., five distinct training instances).

4.6. Numerical Simulations on Data with Missing Values

In this subsection, we will compare the performance of GCNN-nodal (GCNN with only nodal measurements from subsection 4.1), GCNN-hodgeLap and GCNN-edgeLap through numerical simulations on the IEEE 39-bus system and the 57-bus system. GCNN-hodgeLap refers to the GCNN with nodal and line measurements from subsection 4.5 and uses $\bar{\mathbf{S}} = \mathbf{L}_{HG}$ as the line GMD and GCNN-edgeLap refers to the GCNN with nodal and line measurements from the same section but uses $\bar{\mathbf{S}} = \mathbf{L}_{LG}$ as the line GMD. As opposed to earlier simulation results, in this subsection, we test the GCNNs under the scenario where we only have a portion of the measurement data. Therefore, missing values in the input data (i.e., $\mathbf{X}_0$ and $\bar{\mathbf{X}}_0$) are filled with zeros.

The training process is very similar to the one described in subsection 4.2. The only difference is that we now have line measurements and that the partial measurement set is selected randomly for each data point. For the partial measurement data, we consider four levels (100%, 90%, 80%, 70%) for nodal measurements and eight levels (100%, 90%, $\cdots$, 30%) for line measurements. The hyper-parameter values are $K = 2, F_1 = 60, F_2 = 240, F_3 = 120$ for GCNN-nodal and additionally $\bar{K} = 2, \bar{F}_1 = 80, \bar{F}_2 = 320, \bar{F}_3 = 160$ for GCNN-hodgeLap and GCNN-edgeLap. The computational time for training and testing with the IEEE 39-bus system is reported to be less than 10 minutes for GCNN-nodal, and less than 20 minutes for GCNN-hodgeLap and GCNN-edgeLap. The computational time with the IEEE 57-bus system is reported to be less than 15 minutes for GCNN-nodal, and less than 30 minutes for GCNN-hodgeLap and GCNN-edgeLap.

Figure 4 summarizes the RMSE values for these three different methods for the IEEE 39-bus system. Each subfigure corresponds to a different percentage-level (ranging from 100% to 70%) of available nodal measurements and the x-axis of the plot denotes the percentage-level of line measurements. Each point in the subfigure denotes the value of RMSE averaged over five data splits (i.e., five distinct training instances). For GCNN-nodal, since it does not use line measurements, the x-axis (percent of line measurements) is irrelevant. Given a fixed level of nodal measurement data, we can observe that the additional line measurements always results in a more accurate estimate of the system states. For instance in Figure 4(a), we observe that the RMSE values for GCNN-hodgeLap and GCNN-edgeLap are

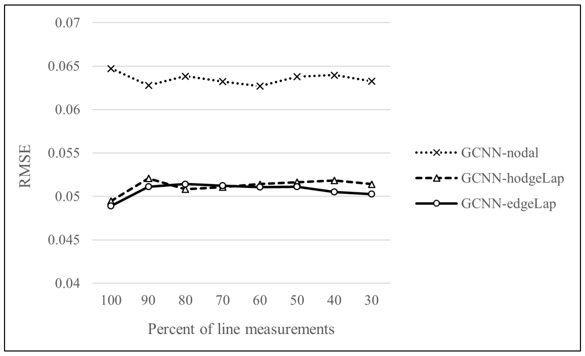


Figure 5: This figure compares the performance of state estimation for the IEEE 57 bus system between three methods: GCNN-nodal, GCNN-hodgeLap and GCNN-edgeLap.

approximately 20 percent less than that of GCNN-nodal. Generally, the level of available line measurements does not seem to have a direct correlation to performance. However, when there is an incomplete set of nodal measurements (less than 100%), having a full set of line measurements drastically performs better than having a partial set of line measurements. This phenomena can be observed in Figures 4(b)-(d). Also, it is observed that there does not exist any significant difference in performance between GCNN-hodgeLap and GCNN-edgeLap.

Figure 5 summarizes the RMSE values for the three different methods for a revised IEEE 57-bus system. We revised the MATPOWER test case by keeping only one line whenever there exists multiple lines between two buses. Note that the RMSE for GCNN-nodal is approximately 0.064, which is higher than the value 0.0393 reported in Table 1. This is due to the fact that we are using a significantly smaller number of parameters for the nodal part of the GCNN architecture. Nevertheless, we observe that the estimation error is small and comparable to optimization based methods. In Figure 6, we also plot the absolute value of the estimation error (for the real part of complex voltage) across all the buses for a sample dataset. Even with an incomplete measurement set, the state estimation errors are small with a maximum error of approximately 0.03.

5. Conclusions

In this paper, we proposed a graph convolutional neural network (GCNN) to *learn* power system state estimators from SCADA and PMU measurement data. We first presented a simple framework for the case of complete nodal measurements. Then, the methodology is expanded so that it is also capable of handling PMU data and line measurements. We observed that even with incomplete measurements, the GCNNs are capable

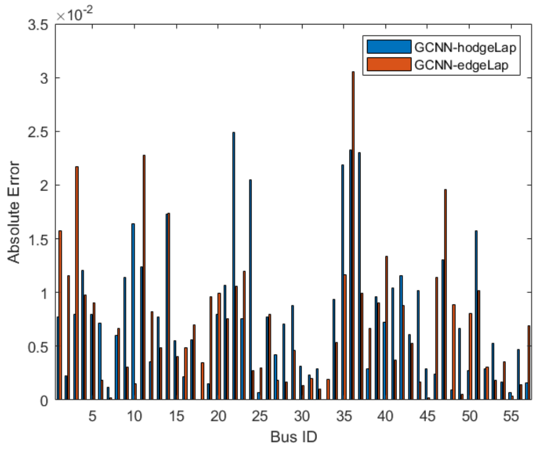


Figure 6: This figure plots the absolute value of the estimation error for the real part of the complex voltages (when using GCNN-hodgeLap and GCNN-edgeLap) with 90% nodal measurements and 90% line measurements. The study is performed on the IEEE 57 bus system and focuses on one specific estimation result.

of producing a high-quality estimate of the underlying voltages. The resulting estimators are distributed and computationally efficient, making them robust to cyber-attacks on the grid and capable of scaling to large networks. Numerical experiments on IEEE test cases verified the efficacy of the proposed method.

References

[1] F. C. Schweppe and J. Wildes, "Power system static-state estimationn, Part I: Exact model," *IEEE Trans. Power Apparatus, Syst.*, vol. PAS-89, pp. 120–125, 1970.

[2] W. W. Kotiuga and M. Vidyasagar, "Bad data rejection properties of weighted least absolute value techniques applied to static state estimation," *IEEE Trans. Power Apparatus, Syst.*, vol. 101, pp. 844–853, 1982.

[3] L. Mili, V. Phaniraj, and P. Rousseeuw, "Least median of squares estimation in power systems," *IEEE PES Summer Meeting*, pp. 493–497, 1990.

[4] L. Mili, M.G. Cheniae, N.S. Vichare, and P.J. Rousseeuw, "Robust state estimation of power systems," *IEEE Trans. Circuits, Syst.*, vol. 41, pp. 349–358, 1994.

[5] M. Göl and A. Abur, "LAV based robust state estimation for systems measured by PMUs," *IEEE Trans. Smart Grid*, vol. 5, pp. 1808–1814, 2014.

[6] W. Xu, M. Wang, J.-F. Cai, and A. Tang, "Sparse error correction from nonlinear measurements with applications in bad data detection for power networks," *IEEE Trans. Signal Process.*, vol. 61, pp. 6175–6187, 2013.

[7] Y. Lin and A. Abur, "Robust state estimation against measurement and network parameter errors," *IEEE Trans. Power Syst.*, vol. 33, pp. 4751–4759, 2018.

[8] S. Park, R. Mohammadi-Ghazi, and J. Lavaei, "Nonlinear least absolute value estimator for topology

error detection and robust state estimation," *IEEE Access*, vol. 9, pp. 137198 – 137210, 2021.

[9] R. Madani, J. Lavaei, and R. Baldick, "Convexification of power flow equations for power systems in presence of noisy measurements," *IEEE Conf. Decision, Control*, pp. 1–8, 2015.

[10] Y. Zhang, R. Madani, and J. Lavaei, "Conic relaxations for power system state estimation with line measurements," *IEEE Trans. Control Network Syst.*, vol. 5, pp. 1193–1205, Sep. 2018.

[11] R. Zhang, J. Lavaei, and R. Baldick, "Spurious local minima in power system state estimation," *IEEE Trans. Control Network Syst.*, vol. 6, pp. 1086–1096, 2019.

[12] L. Ruiz, F. Gama, and A. Ribeiro, "Graph neural networks: Architectures, stability and transferability," *Proc. IEEE*, vol. 109, no. 5, pp. 660–682, May 2021.

[13] M. Jin, H. Chang, W. Zhu, and S. Sojoudi, "Power up! robust graph convolutional network via graph powering," in *35th AAAI Conf. Artificial Intell.*, Sep. 2019.

[14] F. Gu, H. Chang, W. Zhu, S. Sojoudi, and L. El Ghaoui, "Implicit graph neural networks," in *34th Conf. Neural Inform. Process. Syst.*, 2020.

[15] S. Pfrommer, F. Gama, and A. Ribeiro, "Discriminability of single-layer graph neural networks," in *46th IEEE Int. Conf. Acoust., Speech and Signal Process.*, Toronto, ON, 6-11 June 2021, pp. 8508–8512, IEEE.

[16] A. S. Zamzam and N. D. Sidiropoulos, "Physics-aware neural networks for distribution system state estimation," *arXiv:1903.09669v2 [math.OC]*, 12 July 2019.

[17] E. Isufi, F. Gama, and A. Ribeiro, "EdgeNets: Edge varying graph neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, 13 Sep. 2021, early access.

[18] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.

[19] F. Gama, J. Bruna, and A. Ribeiro, "Stability properties of graph neural networks," *IEEE Trans. Signal Process.*, vol. 68, pp. 5680–5695, 25 Sep. 2020.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Adaptive Comput. Mach. Learning. The MIT Press, Cambridge, MA, 2016.

[21] D. Owerko, F. Gama, and A. Ribeiro, "Optimal power flow using graph neural networks," in *45th IEEE Int. Conf. Acoust., Speech and Signal Process.*, Barcelona, Spain, 4-8 May 2020, pp. 5930–5934, IEEE.

[22] D. P. Kingma and J. L. Ba, "ADAM: A method for stochastic optimization," in *3rd Int. Conf. Learning Representations*, San Diego, CA, 7-9 May 2015, pp. 1–15.

[23] M. T. Schaub and S. Segarra, "Flow smoothing and denoising: Graph signal processing in the edge-space," *IEEE Global Conf. Signal and Info. Process (GlobalSIP)*, pp. 735–739, Nov. 2018.