Predicting Power Outages Using Graph Neural Networks

Conference Paper · November 2018

DOI: 10.1109/GlobalSIP.2018.8646486

3 authors:

Damian Owerko
University of Pennsylvania
3 PUBLICATIONS **60** CITATIONS
SEE PROFILE

Fernando Gama
Rice University
76 PUBLICATIONS **772** CITATIONS
SEE PROFILE

Alejandro Ribeiro
University of Pennsylvania
542 PUBLICATIONS **12,073** CITATIONS
SEE PROFILE

Predicting Power Outages Using Graph Neural Networks

Damian Owerko, Fernando Gama and Alejandro Ribeiro

***Abstract*—Power outages have a major impact on economic development due to the dependence of (virtually all) productive sectors on electric power. Thus, many resources within the scientific and engineering communities have been employed to improve the efficiency and reliability of power grids. In particular, we consider the problem of predicting power outages based on the current weather conditions. Weather measurements taken by a sensor network naturally fit within the graph signal processing framework since the measurements are related by the relative position of the sensors. We deploy novel graph neural networks to adequately process weather measurements in order to determine the likelihood of a power outage. Tests on weather measurements taken in the region of New York City show a 1.04% error in the prediction.**

***Index Terms*—Power outages, weather measurements, graph signal processing, graph neural networks**

I. Introduction

Provision of a safe and reliable power source is of utmost importance in modern economies. In fact, the increasing dependence of economic activity on electrical power has prompted scientists and engineers to focus on improving efficiency and reliability of power grids [1]. Likewise, this dependence has turned power outages in a major source of economic loss, which is estimated to be between $22 and $135 billion annually [2], [3]. In many cases, power outages are caused due to harsh weather conditions that affect the distribution infrastructure [3], [4].

Weather conditions are measured by stations located throughout the regions of interest and the measurements collected by different sensors are related by their distance. This lends itself naturally to modeling within the framework of graph signal processing (GSP) [5]. More specifically, the network of weather stations is modeled as a graph on which each node represents a station, and the edges between nodes reflect the distance between these stations. Then, weather measurements taken at each station are modeled as a signal on top of each node, and the underlying graph topology (i.e. the relative position of the stations) is exploited to process these signals [6], [7].

One of the most popular methods for processing data are convolutional neural networks (CNNs) due to their remarkable performance in classification and regression tasks [8], [9]. However, CNNs are designed to operate on regular-structured data such as time series and images, precluding direct application to data given by sensor networks taking weather measurements (i.e. regular CNNs ignore the underlying relationship given by the relative position of the stations). Recently, CNNs have been extended to operate on graph signals [10]–[12], carrying over their success to a new domain.

In this work, we exploit recently developed graph neural networks (GNNs) that act on weather measurements taken at each station and predict power outages in a given region. Weather measurements are taken from Earth Networks historical New York City data [13]. Power outage data for the corresponding region is obtained from EIA Electric Power Monthly [14].

In Section II we introduce the graph neural network architectures to be used and in Section III we formulate the problem and describe it within the graph signal processing framework facilitating the application of the above mentioned GNNs. Numerical experiments on datasets corresponding to the region of New York City are presented in Section IV. Conclusions are drawn in Section V.

Work in this paper is supported by NSF CCF 1717120, ARO W911NF1710438, ISTC-WAS and Intel AI DevCloud. The authors are with the Dept. of Electrical and Systems Eng., Univ. of Pennsylvania. Emails: owerko@sas.upenn.edu, {fgama,aribeiro}@seas.upenn.edu

II. Graph Convolutional Neural Networks

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ be the graph that models the network of weather stations, where $\mathcal{V}$ is the set of N stations, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges connecting stations whose measurements are related, and $\mathcal{W} : \mathcal{E} \to \mathbb{R}$ is a function that reflects the strength of the relationship between the measurements of any pair of stations (typically, a function of the Euclidean distance between the stations' positions). Let $\mathbf{x}^g \in \mathbb{R}^N$ be a graph signal, where $[\mathbf{x}^g]_i$ is a scalar indicating the measurement of weather condition g at node $i \in \mathcal{V}$. Measured weather conditions, indexed by g, include temperature, wind speed, atmospheric pressure, etc. Each vector $\mathbf{x}^g$ is also referred to as a *feature*. The interaction between the graph signal $\mathbf{x}^g$ and the underlying graph support is determined by a matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ called the *graph shift operator* (GSO). The GSO is a matrix that respects the sparsity of the graph, i.e. $[\mathbf{S}]_{ij} = 0$ if $(j, i) \notin \mathcal{E}$ or if $j \neq i$ for any pair of nodes $i, j \in \mathcal{V}$. Examples of GSO commonly used in the literature include the adjacency matrix [6], the Laplacian matrix [5], their normalized counterparts, among many others.

A convolutional neural network acting on graph signals (a graph neural network) aims at obtaining an alternative representation $\mathbf{y}$ of the data $\mathbf{x}^g$ that is useful for the task at hand. This is achieved by a concatenation of L layers, involving a linear operation (referred to as *convolution*), a pooling operation and a point-wise nonlinearity (also known as activation function). Each layer ℓ operates on the incoming $F_{\ell-1}$ features of dimension $N_{\ell-1}$, $\mathbf{x}^g_{\ell-1} \in \mathbb{R}^{N_{\ell-1}}$, $g = 1, \ldots, F_{\ell-1}$, and produce F_ℓ features of dimension N_ℓ, $\mathbf{x}^f_\ell \in \mathbb{R}^{N_\ell}$, $f = 1, \ldots, F_\ell$ as follows

$$\mathbf{x}^f_\ell = \sigma_\ell \left(\mathcal{P}_\ell \left\{ \sum_{g=1}^{F_\ell - 1} \mathbf{H}^{fg}_\ell \mathbf{x}^g_{\ell-1} \right\} \right) \tag{1}$$

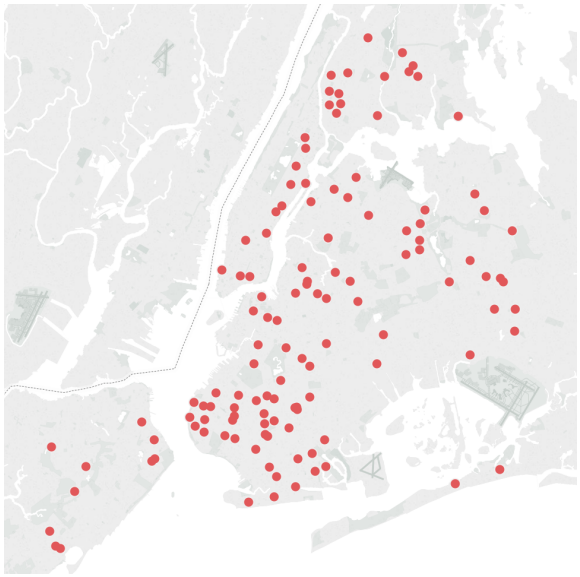

Fig. 1. Location of weather stations in the New York City region.

where σ_ℓ denotes the point-wise nonlinearity, $\mathcal{P}_\ell$ is a pooling operator and $\mathbf{H}_\ell^{fg}$ is a *graph filter* that acts as a convolution.

The graph filter involved in the operation of convolution is

$$\mathbf{H}_\ell^{fg} = \sum_{k=0}^{K_\ell - 1} [\mathbf{h}_\ell^{fg}]_k \mathbf{S}_\ell^{(k)} \tag{2}$$

where each $[\mathbf{h}_\ell^{fg}]_k \in \mathbb{R}$ denotes a filter tap, and $\mathbf{S}_\ell^{(k)} \in \mathbb{R}^{N_{\ell-1} \times N_{\ell-1}}$ is a function of the GSO $\mathbf{S}$ that establishes the pairwise relationship between elements of $N_{\ell-1}$-dimensional feature $\mathbf{x}_{\ell-1}^g$ at layer $\ell - 1$. This implies that the convolution operation (2) preserves locality since only elements related by $\mathbf{S}_\ell^{(k)}$ (which is a function of $\mathbf{S}$) are involved.

The operation of pooling $\mathcal{P}_\ell$ aims at constructing useful summaries as well as reducing the dimensionality of the data to accommodate for a larger number of features, i.e. $N_\ell \leq N_{\ell-1}$ and $F_\ell \geq F_{\ell-1}$. It can be decomposed in a local nonlinearity ρ_ℓ involving nearby neighbors of each node followed by a down-sampling operation $\mathbf{C}_\ell \in \{0,1\}^{N_\ell \times N_{\ell-1}}$ with $N_\ell \leq N_{\ell-1}$, to reduce dimensionality. Different approaches to reducing the dimensionality of the data, and thus obtaining matrices $\mathbf{S}_\ell^{(k)}$ exist in the literature [11], [12], and will be further discussed in Section III.

Finally, we remark that the input to the first layer, $\mathbf{x}_0^g = \mathbf{x}^g$, are the weather measurements taken by each station and that the last layer $\mathbf{x}_L = \mathbf{y}$ yields the appropriate representation. Usually, this last layer is a fully connected layer on which input vector $\mathbf{x}_{L-1}$ collects all F_{L-1} features $\mathbf{x}_{L-1}^g$ and output vector $\mathbf{x}_L$ is thus obtained as a straightforward matrix-vector multiplication $\mathbf{x}_L = \sigma_\ell(\mathbf{A}_L \mathbf{x}_{L-1})$.

One of the hallmarks of GNNs is that they can efficiently learn the architecture parameters from training data. This means that the filter taps $[\mathbf{h}_\ell^{fg}]_k$ and also the linear transform $\mathbf{A}_L$ of the last layer are all obtained by optimizing a given loss function over a training dataset. This is carried out efficiently by means of a back-propagation algorithm [15].

III. Weather Measurements and Power Outages

The dataset [13] contains a collection of weather variables including temperature, sea level pressure, rate of change of pressure, wind speed, among many others. These measurements have been collected from January 2011 to December 2013 throughout the region of New York City, see Fig. 1 for a location of the weather stations. In general, we consider the station network taking weather measurements to have N nodes.

In order to build the underlying graph that relates the measurements across different weather stations, we use the Haversine formula [16] to compute the great circle distance between stations, followed by a Gaussian kernel to compute the weights, and a thresholding operation that keeps only the edges whose weight is above a threshold $\varepsilon_w = 0.01$. More specifically, denote by $d(i,j)$ the great-circle distance between stations i, j given by the Haversine formula,

$$d(i,j) = \tag{3}$$
$$2R \arcsin\left(\sqrt{\text{hav}(\phi_j - \phi_i) + \cos(\phi_i)\cos(\phi_j)\,\text{hav}(\psi_j - \psi_i)}\right).$$

where $\text{hav}(\theta) = \sin^2(\theta/2)$ is the Haversine function, R is the earth's radius, ϕ_i is the latitude of point i and ψ_i is the longitude of point i. This the shortest distance along a sphere's surface, also known as geodesic distance [17]. We then follow by computing weights $\tilde{w}(i,j)$ by means of a Gaussian kernel with $\sigma = 0.1$ applied to the great-circle distance $d(i,j)$,

$$\tilde{w}(i,j) = \exp\left(-\frac{d(i,j)^2}{2\sigma^2}\right). \tag{4}$$

Finally, an edge is drawn between nodes i and j if $\tilde{w}(i,j) > \varepsilon_w$, and the weight $w(i,j)$ associated to that edge is given by $\tilde{w}(i,j)$,

$$(i,j) \in \mathcal{E} \;\Leftrightarrow\; \tilde{w}(i,j) > \epsilon_w$$
$$w(i,j) = \begin{cases} \tilde{w}(i,j) & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}. \tag{5}$$

The GSO adopted is the normalized Laplacian matrix. More precisely, let $\mathbf{W}$ be the weighted adjacency matrix such that $[\mathbf{W}]_{ji} = w(i,j)$ and let $\mathbf{D} = \text{diag}(\mathbf{W}\mathbf{1})$ be the diagonal degree matrix, $[\mathbf{D}]_{ii} = d_i = \sum_{j=1}^N w(i,j)$. Then, the GSO becomes $\mathbf{S} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ which is known as the normalized Laplacian. We consider F_0 input features, given by graph signals $\mathbf{x}^g$, $g = 1, \ldots, F_0$, each of which represents one of the following measurements obtained at each weather station: pressure, temperature, wind speed, pressure rate per hour, humidity, humidity rate per hour and precipitation rate. These graph signals act as an input to the GNN described in (1).

The EIA power grid reliability dataset [14] presents information about the occurrence of power outages in the New York City region during the time period between January 2011 and December 2013. We can thus model each data point y as a binary variable where $y = 1$ means a power outage occurred and $y = 0$ that it did not. Then, the variable y can serve as a label to form pairs $(\mathbf{x}^g, y)$ with the weather measurement and the current state of the power grid (outage or not).

The objective of GNNs (1) is to obtain a representation $\hat{y}$ from data $\mathbf{x}^g$ such that the cross entropy loss between y and $\hat{y}$ is minimized [18]. We consider the following three GNN architectures.

A. Clustering GNN

In the clustering GNN [11], the linear transforms $\mathbf{H}_\ell^{fg}$ are graph filters (2) where in each layer the graph shift operator $\mathbf{S}_\ell^{(k)}$ describes a graph obtained from a multi-scale hierarchical clustering algorithm. More precisely, a new graph $\mathcal{G}_\ell = (\mathcal{V}_\ell, \mathcal{E}_\ell, \mathcal{V}_\ell)$ is associated to the data obtained at the output of each layer ℓ, i.e. $\mathbf{x}_\ell^f$ is a graph signal defined on $\mathcal{G}_\ell$. We adopt a GSO $\mathbf{S}_\ell$ to describe graph $\mathcal{G}_\ell$ and we note that $|\mathcal{V}_\ell| = N_\ell \leq N_{\ell-1} = |\mathcal{V}_{\ell-1}|$, and that $\mathcal{G}_0 = \mathcal{G}$ is the original graph (the original weather station network) with $N_0 = N$. Then, since the convolution operation (2) operates on $\mathbf{x}_{\ell-1}^g$, $\mathbf{H}_\ell^{fg}$ becomes a graph filter defined over $\mathcal{G}_{\ell-1}$, that is $\mathbf{S}_\ell^{(k)} = \mathbf{S}_{\ell-1}^k$. Thus,

the convolution operation consists of a linear combination of k-times shifted versions of input signal $\mathbf{x}^g_{\ell-1}$ on the underlying graph $\mathcal{G}_{\ell-1}$. Pooling $\mathcal{P}_\ell$ is carried out using a binary tree structure of the data which matches the reduction in size $N_{\ell-1}/N_\ell$ of the graphs on each layer. See [11] for more details.

In this architecture, the hyper-parameters are the number of layers L, as well as the number of features F_ℓ and the number of filter taps K_ℓ at each layer. Also, the choice of the multi-scale hierarchical clustering and the reduction in size $N_{\ell-1}/N_\ell$ of the graph between each layer have a strong impact on the performance of the algorithm.

B. Selection GNN

The selection GNN [12] bypasses the need to generate new graphs on each layer by operating directly on the original graph (the given network of weather stations). This is achieved by means of down-sampling and zero-padding. On each layer ℓ a smaller number of nodes are selected $N_\ell \leq N_{\ell-1}$ by using selection matrix $\mathbf{C}_\ell \in \{0,1\}^{N_\ell \times N_{\ell-1}}$. This implies that for each layer, feature $\mathbf{x}^f_\ell$ is only defined at the selected N_ℓ nodes. In order to be able to employ a graph filter defined on the original graph, we zero-pad the input to the graph filter to make it fit the original graph, we filter it, and then we down-sample it again to the nodes on which it was originally defined. More specifically, let $\mathbf{D}_\ell = \mathbf{C}_\ell \mathbf{C}_{\ell-1} \cdots \mathbf{C}_1 \in \{0,1\}^{N_\ell \times N}$ be the selection matrix that determines the N_ℓ nodes kept at layer ℓ out of the original N nodes. Then, the convolution operation by means of a graph filter becomes

$$\mathbf{H}^{fg}_\ell = \mathbf{D}_{\ell-1} \left(\sum_{k=0}^{K_\ell - 1} [\mathbf{h}^{fg}_\ell]_k \mathbf{S}^k \right) \mathbf{D}^\mathsf{T}_{\ell-1} \mathbf{x}^g_{\ell-1} \tag{6}$$

where $\mathbf{D}^\mathsf{T}_{\ell-1}\mathbf{x}^g_{\ell-1}$ zero-pads the input feature so that the original GSO $\mathbf{S}$ can be applied, and then multiplying by $\mathbf{D}_{\ell-1}$ keeps the output of the graph filter only at the same nodes on which the input feature $\mathbf{x}^g_{\ell-1}$ was defined. In this case, the shift matrix in (2) is $\mathbf{S}^{(k)}_\ell = \mathbf{D}_{\ell-1}\mathbf{S}^k\mathbf{D}^\mathsf{T}_{\ell-1}$. In the selection GNN, pooling $\mathcal{P}_\ell$ is carried out by means of a summarizing function ρ_ℓ that acts on the α_ℓ-hop neighborhood of each node

$$\left\{ j : [\mathbf{S}^{(k)}_\ell]_{ij} \neq 0, \text{ for some } k \leq \alpha_\ell \right\}. \tag{7}$$

More details can be found in [12, Section III].

When employing this architecture, the hyper-parameters are also the number of layers L, and the number of features F_ℓ and filter taps K_ℓ at each layer. Additionally, the number of nodes to select on each layer N_ℓ as well as the method for selecting them have a big impact on the performance of the algorithm. Finally, the size of the pooling neighborhood α_ℓ is also another hyper-parameter.

C. Aggregation GNN

Aggregation GNNs follow quite a different philosophy than the previous two architectures [12], based on the notion of aggregation sampling [19]. In this case, a single node $p \in \mathcal{V}$ is selected, and information is gathered at that node by means of successive exchanges with the neighboring nodes. That is, at the first stage of the aggregation architecture input measurements $\mathbf{x}^g$ are gathered at a single node by means of successive applications of the shift

$$\mathbf{z}^g(p,N) = \left[[\mathbf{x}^g]_p, [\mathbf{S}\mathbf{x}^g]_p, \ldots, [\mathbf{S}^{N-1}\mathbf{x}^g]_p \right]^\mathsf{T}. \tag{8}$$

The resulting signal $\mathbf{z}^g(p,N)$ collected at node p after N neighbor exchanges exhibits a regular structure, analogous to time series, since consecutive elements in the vector actually relate neighboring values of the signal. Now, given that signal $\mathbf{z}^g(p,N)$ is regular, we can use it as input $\mathbf{z}^g_0 = \mathbf{z}^g(p,N)$ to a regular CNN: at each layer ℓ, the convolution operation becomes a regular convolution between filter taps $[\mathbf{h}^{fg}_\ell]_k$ and input $\mathbf{z}^g_{\ell-1}$, and pooling becomes regular pooling. See [12, Section IV] for details.

Algorithm 1 Greedy algorithm for station selection.

Input: $\mathcal{S}$: set of weather stations, N: number of stations to select
$\mathbf{A}$: $[\mathbf{A}]_{ij} = 1$ if data at time i in station $j \in \mathcal{S}$ is available
Output: $\mathcal{C}$: selected weather stations

procedure GREEDY_SELECTION($\mathcal{S}$,N,$\mathbf{A}$)
  Set $\mathcal{C} = \emptyset$
  while $|\mathcal{C}| < N$ **do**
    $n = \text{argmax}_{k \in \mathcal{S} \setminus \mathcal{C}} \{ \sum_i \mathbf{1}\{[\mathbf{A}]_{ij} = 1, \forall\ j \in \mathcal{C} \cup \{k\}\}\}$
    $\mathcal{C} = \mathcal{C} \cup \{n\}$
  end while
end procedure

In the aggregation architecture, the most important hyper-parameter becomes the selected node p to aggregate all the information and construct vector $\mathbf{z}^g(p,N)$. Once vector $\mathbf{z}^g(p,N)$ is obtained, the hyper-parameters are the same as any regular CNN: the number of layers L, the size K_ℓ and stride of the convolution operation, the number of features F_ℓ and the size of the pooling function α_ℓ.

The aggregation architecture can be tweaked to avoid the large number N of exchanges required to build the regular vector $\mathbf{z}^g(p,N)$. This leads to the multi-node aggregation architecture [12]. In this case, instead of selecting a single node p, we select a subset of nodes $\mathcal{P} \subset \mathcal{V}$. These nodes gather neighboring information by means of $Q \ll N$ exchanges, building a collection of regular vectors but of smaller size (of size Q instead of size N) given by $\mathbf{z}^g(p,Q)$, for $p \in \mathcal{P}$. A regular CNN can be applied on each of these nodes to obtain features describing the neighborhood. The regular CsNN at each node are called inner layers and mimic the aggregation architecture. Once this features are built on each node, they can be exchanged with the other nodes by means of zero-padding to build summaries of larger neighborhoods. Each of these further exchanges is called an outer layer. Details can be found in [12, Section IV.A].

For the multi-node variation of the aggregation architecture, the hyper-parameters are the subset of nodes $\mathcal{P}$ selected to aggregate the neighboring information and the number Q of exchanges. Also, the hyper-parameters of the regular CNN run on each node.

IV. Numerical experiments

We draw the data for the numerical experiments from two datasets [13], [14]. First, dataset [13] contains hourly logs registering weather measurements from 123 stations in the New York City region (see Fig. 1). These hourly logs run from January 2011 to December 2013, registering $26,304$ data points. Second, dataset [14] is a chronological list of major electrical grid disturbance events, registering the start time and duration of the corresponding event. In the time period being considered 25 such events occurred, of which 18 were caused by severe weather and 6 by fuel supply emergencies. On average $185,872$ consumers were affected by these disturbances.

Preprocessing is needed to make both datasets compatible, and this is described in detail in Section IV-A. The specifics of the graph neural network architectures employed can be found in Section IV-B. For comparison with baseline methods, we consider fully connected neural networks and affine space models in Section IV-C. Results and discussion is in Section IV-D.

TABLE I: Selected GNN results across experiments. Parameters follow the notation from section II.

Architecture	Clustering	Selection	Aggregation	Multi-node	No pooling	Neural network	NN with sample weights	PCA
F1 score	68.18	48.28	64.86	74.42	**86.36**	0.0	61.47	50.70
Error(%)	2.42	2.60	2.25	1.90	**1.04**	3.81	2.88	4.79

A. Preprocessing

In order to make the two datasets compatible, and to model them within the GSP framework described before, some preprocessing is needed. We consider each hourly measurement registered in [13] to be a data point in the dataset. A label y is associated to each hourly measurement by checking in dataset [14] if, at that hour, a major electrical grid disturbance event is registered. A label $y = 1$ signals that a disturbance occurred and a label $y = 0$ that it did not. In this way, we create labeled data points $(\mathbf{x}^g, y)$ where each $[\mathbf{x}^g]_i$ is the value obtained for weather measurement g at station i.

We find that in [13] there is a considerable amount of missing data. More specifically, not all weather stations have registered all measurements at all times. Therefore, we need to carefully select a subset of weather stations which have registered the weather measurements for as many hours as possible. To do this we implement a greedy selection algorithm that maximizes the amount of available data. This procedure is described in algorithm 1.

Finally, we get a dataset of $5,777$ data points, where each point consists of weather measurements taken at $N = 25$ weather stations. All studied weather measurements are present at all of these 25 weather station for all $5,777$ hourly logs, meaning that there is no missing data within this dataset. We also note that, for these $5,777$ logged hours there were 218 major power disturbance events registered. A total of 7 weather measurements from each station at each hour were used: temperature, wind speed, pressure rate per hour, humidity, humidity rate per hour and precipitation rate. A test set is built by randomly choosing 10% of the data points in the dataset, and the rest are used to constitute the training set.

B. Graph Neural Networks

The GNN architectures under study are the clustering (Section III-A), selection (Section III-B), aggregation and multi-node (Section III-C) ones, all of them consisting of $L = 2$ layers. Additionally, we consider a GNN architecture (1) in which there is no pooling (i.e. there are two layers, each of which keeps all the original nodes).

All architectures are trained using the ADAM optimizer, with a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ [20]. The batch size used is 100. The hyper-parameters are determined by exploring the hyper-parameters space, choosing those that maximize the F1 score and the accuracy.

In particular, for the clustering GNN we obtain $F_1 = 14$ and $F_2 = 28$ features in each layer, with filters of length $K_1 = 7$ and $K_2 = 14$. At each layer, we reduce the number of nodes by half. For selection GNN, we keep 16 nodes in the first layer and 8 nodes in the following one. The nodes kept are those with highest degree. We obtain $F_1 = F_2 = 16$ features with filters of length $K_1 = K_2 = 16$ and pooling reaches out to the $\alpha_1 = \alpha_2 =$ 2-hop neighborhood. In aggregation GNN, we select the node p with the highest degree, and then once we compute vector $\mathbf{z}^g(p, N)$ in (8), we proceed with a regular CNN that computes $F_1 = F_2 = 16$ features at each layer with filters with $K_1 = K_2 = 16$ taps each and pooling of size $\alpha_1 = \alpha_2 = 2$. The multi-node version has 2 outer layers in which all nodes are kept first and 10 nodes are selected later and where $Q_1 = 15$ neighboring exchanges are carried out in the first layer and $Q_2 = 10$ in the second one. Then, once in each node, a regular CNN with 2 layers is deployed that extracts 16 features with filters of size $K = 8$ and pooling size 2. Finally, the GNN without pooling computes $F_1 = 14$ and $F_2 = 28$ features on each layer, with filters having $K_1 = 7$ and $K_2 = 14$ taps.

C. Baseline methods

To benchmark against classical neural networks, we train a network with L fully connected layers with F_ℓ hidden units on each layer, a ReLU activation function and dropout with probability $1 - d$. The weights were initialized uniformly between -0.05 and 0.05. The network input is a concatenation of the seven weather measurements $\mathbf{x}^g$, $g = 1, \ldots, 7$, that is $\mathbf{x} = [(\mathbf{x}^1)^\mathsf{T}, \ldots, (\mathbf{x}^7)^\mathsf{T}]^\mathsf{T}$, for each of the N stations with a total length of length $7N = 175$, since $N = 25$ as before. Each of the 7 input features was normalized so that measurements are in $[0, 1]$. We also train the network using the ADAM optimization algorithm, with learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ with a batch size of 32 over 1000 epochs. After an exhaustive search in the hyper-parameters space, we select $L = 1$ layers, $F_1 = 1,000$, and $d = 0$.

We noted that, during initial experiments using neural networks the model would quickly converge to a trivial solution where the output was always negative. This tends to happen due to the large proportion of $y = 0$ labels. In fact, if the neural network (or any other estimation method) would predict 0 for all input features, then the error would be 3.77%. In order to overcome this, we weighted more heavily errors in samples with positive labels. This implies the introduction of another hyper-parameter that represents the relative weight of the positive samples. We found that reweighing samples with positive labels 10 times more than samples with negative labels led to best results. This had the desired effect of preventing a trivial solution and therefore led to substantial improvement in F1 score (an F1 score of 0 indicates that there is a 100% missed detections). Both results (with and without reweighing) are presented in Section IV-D for comparison.

Finally, we also consider a baseline using an affine space model [21]. In this case, we decide to use only one feature, sea level pressure measurements, since we observed that using more features decreased the performance. Then, we estimate the intraclass mean $\boldsymbol{\mu}_y$ and covariance matrix $\boldsymbol{\Sigma}_y$, and project the sea level pressures measurements $\mathbf{x}$ on the eigenvectors of each covariance matrix. Finally, we assign a class to that one whose eigenvector matrix minimizes the projection. More precisely,

$$\hat{y} = \underset{y \in \{0,1\}}{\operatorname{argmin}} ||\mathbf{V}_y(\mathbf{x} - \boldsymbol{\mu}_y)||. \tag{9}$$

D. Results

As performance measures, we use the F1 score [22], which highlights the fact that the number of positive labels is small, and the error percentage, counted simply as the total number of prediction errors over the total number of predictions. Results are shown in Table. I.

The best performing architecture, both in terms of F1 score and error is the GNN with no pooling, closely followed by the multi-node GNN. We observe that an error of 1.04% is an improvement of

70% with respect to the baseline of 3.77% error (obtained by just estimating no power outage irrespective of the weather measurements).

We also see in Table I that all GNN architectures outperform, not only the baseline error, but also the one obtained with the re-weighted fully connected neural network. This is also true in terms of the F1 score (with the exception of selection GNN).

V. Conclusions

In this work, we have addressed the problem of predicting power outages based on weather measurements. Given the natural description of the problem in terms of the graph signal processing framework, and the success of neural networks, we proceeded to deploy novel graph neural networks. More specifically, we modeled weather measurements and graph signals and used these as features to feed five different GNN architectures. Results show that all of these architectures improve over the baseline errors, including fully connected neural networks. In particular, the best performing architecture bring a 70% improvement in the prediction error.

References

[1] K. Moslehi and R. Kumar, "A reliability perspective of the smart grid," vol. 1, no. 1, June 2010, pp. 57–64.

[2] K. H. LaCommare and J. H. Eto, "Cost of power interruptions to electricity consumers in the United States (U.S.)," *Lawrence Berkeley National Laboratory*, no. LBNL-58164, Feb. 2006.

[3] R. J. Campbell, "Weather-related power outages and electric system resiliency," Congressional Research Service, CRS Report for Congress 7-5700 (R42696), 28 Aug. 2012.

[4] M. Panteli and P. Mancarella, "Influence of extreme weather and climate change on the resilience of power systems: Impacts and possible mitigation strategies," *Electric Power Systems Research*, vol. 127, pp. 259–270, Oct. 2015.

[5] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[6] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.

[7] A. Sandyhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042–3054, June 2014.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional neural networks for large-scale image recognition," *arXiv:1409.1556v6 [cs.CV]*, 10 Apr. 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[9] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *13th European Conf. Comput. Vision 2014*, Zürich, Switzerland, 6-12 Sep. 2014.

[10] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and deep locally connected networks on graphs," *arXiv:1312.6203v3 [cs.LG]*, 21 May 2014. [Online]. Available: http://arxiv.org/abs/1213.6203

[11] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Neural Inform. Process. Syst. 2016*. Barcelona, Spain: NIPS Foundation, 5-10 Dec. 2016.

[12] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *arXiv:1805.00165v1 [eess.SP]*, 1 May 2018. [Online]. Available: http://arxiv.org/abs/1805.00165

[13] Earth Networks, "Historical observations," 13 March 2018. [Online]. Available: http://wcai.wharton.upenn.edu/earth-networks-data-portal/

[14] U.S. Energy Information Administration, "Electric power monthly," 25 March 2014. [Online]. Available: https://www.eia.gov/electricity/monthly/

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[16] R. W. Sinnott, "Virtues of the haversine," *Sky and Telescope*, vol. 68, no. 2, p. 159, Dec. 1984.

[17] K. Gade, "A non-singular horizontal position representation," *J. Navigation*, vol. 63, no. 3, pp. 395–417, July 2010.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ser. The Adaptive Computation and Machine Learning Series. Cambridge, MA: The MIT Press, 2016.

[19] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Sampling of graph signals with successive local aggregations," *IEEE Trans. Signal Process.*, vol. 64, no. 7, pp. 1832–1843, Apr. 2016.

[20] D. P. Kingma and J. L. Ba, "ADAM: A method for stochastic optimization," in *3rd Int. Conf. Learning Representations*. San Diego, CA: Assoc. Comput. Linguistics, 7-9 May 2015.

[21] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, Aug. 2013.

[22] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation," *J. Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.