Cascading Failure Prediction in Power Grid Using Node and Edge Attributed Graph Neural Networks

Karuna Bhaila[1], **Xintao Wu**[1]

[1]Department of Electrical Engineering and Computer Science
University of Arkansas
{kbhaila, xintaowu}@uark.edu

Abstract

In recent years, probabilistic data-driven methods have been gaining popularity for the study of cascading failures in power systems. These data-driven models are well-suited for analysis of large exploration spaces, historical event data, and online cascade prediction compared to computationally expensive physics-based methods. Motivated by the nature of failure propagation during cascade events, in this work, we propose using Graph Neural Networks (GNNs) to study cascading failures in power grids initiated due to contingencies introduced to the network's profile. The goal is to train GNN models using network profiles modified to simulate contingencies of varying sizes and predict the vulnerability of buses and branches after the cascade event has stabilized. We empirically verify the performance of several GNN models on these tasks. We also formulate a graph convolution mechanism to perform message-passing between power grid components and effectively utilize both bus and branch features. Our experimental evaluation demonstrates the efficiency of GNN-based methods for end-to-end cascading failure prediction on the IEEE 39-bus and 118-bus test systems.

Introduction

Graph data representation has been gaining wide traction in various scopes within natural sciences due to its adaptability in modeling complex and structured data. Simultaneously, researchers in these fields have also adopted Graph Neural Networks for predictive and analytical purposes on such data (Gilmer et al. 2017; Sanchez-Gonzalez et al. 2020). Likewise, graph representation is applicable for data modeling in power systems as their grid structures can naturally be formulated as graphs preserving their non-euclidean properties. This has motivated the use of GNNs in power systems studies for predictive, diagnostic, and mitigative purposes. For instance, GNNs have recently been used for power flow calculations, cascading failure event analysis, outage prediction, time series prediction, fault diagnosis, and so on (Liao et al. 2021).

Cascading failure refers to the mechanism of failure propagation in physical systems triggered by a set of initial disturbances causing subsequent failures in other system components (Dobson et al. 2007). In power systems, cascading failures may be triggered by weather-related events, natural disasters, personnel errors, overloading, line or generator outages, or instability (Vaiman et al. 2012). These initial disturbances may cause small-scale outages or widespread blackouts due to failure propagation. Grid failures may also affect operations in other energy networks due to the coupling of such infrastructure networks with power systems. In any case, many services are interrupted, adversely affecting society and/or the economy at large. For the prevention, detection, and mitigation of such disastrous events, researchers have largely emphasized the study of cascading failures in power systems.

Cascading failure prediction plays a major role in the prevention of cascade events as the mechanisms can be used to identify and localize high-risk areas and subsequently reinforce protective measures ahead of time. With the increased availability of outage data based on cascading failure models, researchers have focused on developing machine learning-based algorithms to predict cascading failures. For example, Shuvro et al. (2019) utilize power grid operating parameters and cascading failure model features to predict cascading failure size using various linear classifiers such as logistic regression, k-nearest neighbors, decision tree, random forest, support vector machine, and AdaBoost. For further granularity, Zhang et al. (2021) utilize topology and power flow information to evaluate bus vulnerability using an XGBoost classifier. Liu et al. (2021) employ Graph Convolutional Network (GCN) (Kipf and Welling 2017) within a physics-based cascade path search algorithm and predict branch failure probability using the network state at each iteration of the search algorithm and guide the path search with GNN predictions. Varbella, Gjorgiev, and Sansavini (2023) investigate the use of GNNs for the task of predicting a graph-level binary value of whether the network conditioned on a set of single or multiple component failures will result in the load demand not being served at the end of the cascade event.

In this work, we investigate a similar use of GNNs for end-to-end cascading failure prediction. However, in contrast to the network-level classification tasks studied in earlier works with GNNs, we aim to predict the effect of initial contingencies on individual components of the power network. This is a comparatively more fine-grained task that can provide additional insights regarding failure propagation and identify critical regions in the power system. This knowledge allows

network operators to implement preventive strategies and reduce the risk of blackouts (Vaiman et al. 2012). To the best of our knowledge, this is the first work to explore the use of GNNs for end-to-end prediction of the effects of cascading failure for buses and branches simultaneously. Although Liu et al. (2021) explore the use of the GCN in predicting line outage, their method uses the GCN as a guide to speed up the search for cascading failure paths within a physics-based search model. In our framework, we rely only on the network profile containing initial contingencies to make the final predictions rather than iteratively stepping through the cascade process to find failure paths. This approach reduces the computational overhead of calculating multiple power flow solutions in each iteration. We achieve this by modeling graph convolutions that mimic failure propagation by means of message-passing among the network components. The propagated messages encode information regarding the operational status of the buses and branches conditioned on the contingencies. From the graph convolutions, we obtain vector representations for the components that can be used as input for downstream classifiers to categorize their status as *fail* or *operational* at the end of the cascade event. We train the model to simultaneously predict the failure probability of both buses and branches. The trained model can thus be deployed to detect faults in power systems before catastrophic events occur.

We summarize our main contributions as follows:

- We formulate **N**ode and **E**dge **A**ttributed **GNN** (NEA-GNN), a graph convolution method that can effectively utilize bus and branch attributes and map both components into a latent embedding space simultaneously.
- We propose a general framework to train GNN models using cascading failure data and predict the vulnerability of grid components in a data-driven and end-to-end manner.
- We generate a cascading failure dataset using an AC-based cascade simulation model and empirically evaluate various GNN methods and the proposed convolution approach. The experimental results show that the proposed method is effective for vulnerability prediction of power grid components.

Methods

Data Collection and Modeling

Data-centric approaches in machine learning generally require a large amount of data for training. However, historical occurrences of cascading failures are infrequent (Guo et al. 2017) and far too few to train ML-based methods. Nonetheless, the development of cascade simulation models for power grids (Dobson, Carreras, and Newman 2005; Pambour et al. 2017; Song et al. 2016; Noebels, Preece, and Panteli 2022) has made it possible to generate large-scale data for resilience studies and cascading failure analysis. These simulation methodologies model various physical mechanisms of real-world cascading failures in power systems. The models may consider dynamic or steady-state systems and assume different operating conditions and power flow mechanisms. In this work, we utilize an AC-based cascading failure simulation model, AC-CFM (Noebels, Preece, and Panteli 2022) to generate synthetic simulation data. AC-CFM is a Matlab-based cascading failure model that has been validated using recommendations of the IEEE PES working group (Vaiman et al. 2012). The simulation model incorporates dynamic phenomena and protection mechanisms using static representations and handles cascades recursively. AC-CFM also handles non-convergence in AC power flows by disabling line constraints, disabling voltage limits, and making all loads dispatchable. This makes the simulation model stable for very large contingencies. For more details regarding the simulation model, we refer readers to the reference (Noebels, Preece, and Panteli 2022).

For a given power system, we first define a default state of the network that is fully operational and does not trigger any cascade events. For a network with N branches, a scenario is obtained by simulating $k < N$ branch failures as the initial contingencies;v k is chosen such that cascade scenarios with small as well as very large contingencies are generated. For each scenario, we first randomly sample k, the number of initial failures. Then, the first failure component is randomly picked from all branches in the network. The subsequent failures are then sampled such that they are connected to at least one contingency sampled before them. This allows us to simulate real-life failure events where component failures may occur geographically close to each other. The AC-CFM model incorporates the initial set of contingencies into the network by disabling their status before the start of the simulation. The model then runs a cascade simulation involving recursive computations of power flows and applications of protection mechanisms. Once the cascade halts, AC-CFM outputs the status of the entire network and its components. We utilize these final operational statuses of buses and branches as the target labels for the corresponding scenario. For bus and branch features, we utilize the network state parameters and initial power flow solutions of the scenario.

To model the simulation data for graph-based approaches, we represent each scenario as a directed and unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ denotes the node set corresponding to the buses and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ denotes the edge set corresponding to the branches. Additionally, we represent bus features using a matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times D_{\mathcal{V}}}$ and branch features as $\mathbf{Z} \in \mathbb{R}^{|\mathcal{E}| \times D_{\mathcal{E}}}$; $D_{\mathcal{V}}$ and $D_{\mathcal{E}}$ refer to the number of bus attributes and branch attributes respectively. Echoing the representation of contingencies in AC-CFM, we include branch operational status at the start of the cascade as a binary attribute in $\mathbf{Z}$ to denote whether a branch is included in the contingency set of that scenario. Furthermore, label vector $Y = \{0, 1\}^{|\mathcal{V}|}$ represents the post-cascade status of all buses in the network, where $y_v = 0$ indicates that bus v was tripped during the cascade and $y_v = 1$ denotes that v is operational after the cascade. We define a similar label vector for all edges in the network as $F = \{0, 1\}^{|\mathcal{E}|}$ where $f_e = 0(1)$ denotes the *fail* (*operational*) status of branch e. We further define an incidence set $\mathcal{I} \in \mathcal{V} \times \mathcal{E}$ containing node-edge pairs to represent the bus and branch incidence relationship of the network. We use $\mathcal{I}(v)$ to denote the neighborhood of v in $\mathcal{I}$, i.e., the edges incident on v and $\mathcal{I}(e)$ to mean the set containing the source and sink nodes of e. Our goal here is to train a

graph-based model using bus features $\mathbf{X}$ and branch features $\mathbf{Z}$ that reflect the initial state of the network conditioned on the contingency set, along with the grid topology, to compute bus and branch embeddings and ultimately predict their final operational states $\hat{Y}$ and $\hat{F}$.

Graph Neural Networks

Motivated by the success of convolutional neural networks in the image domain, GNNs were developed to extend convolutions to the graph domain (Wu et al. 2020). Primarily, GNNs learn vector representations for a target node in a graph through iterative information propagation in its neighborhood. The learned representations can be used for various supervised and unsupervised downstream tasks. Based on the mechanisms of propagation, GNNs can be broadly categorized into spectral-based and spatial-based methods. Spectral GNNs define graph convolutions based on spectral graph theory and use the graph Laplacian for information propagation. Spatial GNNs operate in local neighborhoods to update node representation as an aggregate of its neighbors' representations. Generally, a k-layer GNN consists of k sequential graph convolutional layers that implement message passing and neighborhood aggregation to update node representations. The updating process of the k-th layer in GNN can be represented as

$$\begin{aligned} \mathbf{h}^k_{\mathcal{N}(v)} &= \mathrm{AGG}^k(\{\mathbf{h}^{k-1}_u,\ \forall u \in \mathcal{N}(v)\}), \\ \mathbf{h}^k_v &= \mathrm{UPDATE}^k(\mathbf{h}^k_{\mathcal{N}(v)}; \mathbf{W}^k), \end{aligned} \tag{1}$$

where $\mathbf{h}^k_v$ is node v's representation at the k-th layer, $\mathcal{N}(v)$ is its neighborhood set obtained from $\mathcal{E}$. $\mathrm{AGG}^k(\cdot)$ is a differentiable aggregator function and $\mathrm{UPDATE}^k(\cdot)$ is a non-linear function with learnable parameters $\mathbf{W}^k$ of the k-th layer.

GCN (Kipf and Welling 2017) is one of the earliest and most widely used GNN methods. GCN is a spectral convolution approach and uses a first-order approximation of the graph Laplacian to update node embeddings in each layer. GraphSAGE (Hamilton, Ying, and Leskovec 2017) is a spatial-based method that updates node representations based on aggregates from the node's sampled neighborhood in an inductive manner. Both GCN and GraphSAGE do not incorporate edge features when updating node representations and assign equal importance to all neighbors of a node for aggregation. Graph Attention Network (GAT)(Velickovic et al. 2018) is another spatial-based method and adopts the attention mechanism to learn edge-specific weights using node and/or edge features during convolution. Graph Isomorphism Network (GIN) (Xu et al. 2019) is another class of spatial GNN developed to achieve maximum expressivity in GNNs. GIN learns to distinguish different graph structures by learning different embeddings to represent non-isomorphic graphs. Similar to GCN and GraphSAGE, GIN also uses only node features in its update function.

Although some of the GNNs above may utilize edge features, the models were primarily designed to update node embeddings. The node embeddings from the final layer can be used as input for a feed-forward neural network to obtain the node status predictions as $\hat{y}_e$ for all $v \in \mathcal{V}$. Moreover, in failure propagation, the tripped branches are likely incident on or at least in the vicinity of the tripped buses. Therefore we can simply utilize the dot product of embeddings of node pairs to quantify the probability of failure of the edge between them as follows:

$$\hat{f}_e = \mathbf{h}_u \cdot \mathbf{h}_v \tag{2}$$

where u and v are the nodes at the endpoint of edge e.

Node and Edge Attributed Graph Convolutions

It is reasonable to assume that incorporating edge features during training can help the model learn better embeddings since these features may contain meaningful information correlated with target variables. Simply using these features to compute edge weights as in GAT may be inadequate and cause loss of information when reducing multi-dimensional features to a scalar value. This could affect model performance, especially for edge-related prediction tasks.

Here, we propose to utilize edge attributes during graph convolution and implement message-passing between node pairs as well as node-edge pairs. We refer to this modified convolutional approach as NEA-GNN. Our formulation of NEA-GNN incorporates the use of edge representations during message-passing similar to the graph convolutions used in (Kearnes et al. 2016) where information is exchanged between nodes and edges. In NEA-GNN, node pairs exchange and aggregate information using their neighbors' representations similar to any standard GNN. Additionally, we define weights in the convolution layer to linearly transform the edge features. Further, to facilitate information propagation between nodes and edges, we implement message-passing between these components using their incidence relations. In other words, we update node representations as a combination of the aggregated node neighbors' representations and the aggregated edge neighbors' representations. We also update edge representations by combining their linear transformations with aggregated node neighbors' representations. This bidirectional flow of information results in implicit message-passing between the edges of the graph.

However, in small or very sparse graphs, such an approach might inadvertently result in over-smoothing of the embeddings even during short-range propagation. Motivated by the use of alternating node and edge embeddings update mechanism in CensNet (Jiang, Ji, and Li 2019), we propose to append node-edge and edge-node message-passing mechanisms to the convolution layer in turns. In odd-numbered layers, we update edge embeddings using only edge features whereas we update node embeddings using node features as well as propagated information from edges as shown below:

$$\mathbf{h}^k_e = \mathbf{W}^k_1 \cdot \mathbf{h}^{k-1}_e, \tag{3}$$

$$\mathbf{h}^k_{\mathcal{I}(v)} = \mathrm{AGG}^k\left(\{\mathbf{h}^k_e, \forall e \in \mathcal{I}(v)\}\right), \tag{4}$$

$$\mathbf{h}^k_v = \sigma\left[\mathbf{W}^k_2 \cdot \{\mathbf{h}^{k-1}_v \cup \mathbf{h}^k_{\mathcal{N}(v)}\} || \ \mathbf{W}^k_3 \cdot \mathbf{h}^k_{\mathcal{I}(v)}\right], \tag{5}$$

where $||$ refers to the concat operation, $\mathcal{I}(v)$is the edge neighborhood of v, and $\mathbf{h}^0_e$ and $\mathbf{h}^0_v$ correspond to the branch features $\mathbf{z}_e$ and bus features $\mathbf{x}_v$ respectively. In even-numbered layers, node representations are updated using only node

neighbors' embeddings whereas the edges additionally receive information propagated from connected nodes. We formulate this type of convolution as

$$\mathbf{h}_v^k = \mathbf{W}_2^k \cdot \{\mathbf{h}_v^{k-1} \cup \mathbf{h}_{\mathcal{N}(v)}^k\}, \tag{6}$$

$$\mathbf{h}_{\mathcal{I}(e)}^k = \mathrm{AGG}^k\left(\{\mathbf{h}_v^k, \forall v \in \mathcal{I}(e)\}\right), \tag{7}$$

$$\mathbf{h}_e^k = \sigma\left[\mathbf{W}_1^k \cdot \mathbf{h}_e^{k-1} || \ \mathbf{W}_4^k \cdot \mathbf{h}_{\mathcal{I}(e)}^k\right], \tag{8}$$

where $\mathcal{I}(e)$ denotes the node neighborhood of e. The node and edge embeddings are updated in each convolutional layer but the information is propagated between them in an alternating manner. From the final convolution layer, we obtain embeddings for both nodes and edges that can be utilized for any downstream task by feeding them into a task-specific learning model. For cascading failure prediction, we train separate feed-forward neural network classifiers for each component to predict their respective failure status. We obtain the predicted probabilities $\hat{f}_e$ and $\hat{y}_v$ from the classifiers using $\mathbf{h}_e$ and $\mathbf{h}_v$ as inputs respectively. Algorithm 1 shows this process for obtaining node and edge predictions via alternating convolutions for a single graph.

Algorithm 1: Forward pass using NEA-GNN

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, node features $\mathbf{X}$, edge features $\mathbf{Z}$, node labels Y, edge labels F, number of layers K, differentiable aggregator functions AGG^k, $\forall k \in \{1, \ldots, K\}$, weight matrices W_1^k, W_2^k, W_3^k and W_4^k $\forall k \in \{1, \ldots, K\}$, classifiers $\mathrm{MLP}_{\mathcal{V}}$ and $\mathrm{MLP}_{\mathcal{E}}$

Output: Bus predictions $\hat{y}_v$ for all $v \in \mathcal{V}$ and branch predictions $\hat{f}_e$ for all $e \in \mathcal{E}$

▷ NEA-GNN for node and edge embeddings

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v \ \forall v \in \mathcal{V}$
$\mathbf{h}_e^0 \leftarrow \mathbf{z}_e \ \forall e \in \mathcal{E}$
for $k = 1, \ldots, K$ **do**
  if k is odd **then**
    $\mathbf{h}_e^k \leftarrow \mathbf{W}_1^k \cdot \mathbf{h}_e^{k-1}, \ \forall e \in \mathcal{E}$
    $\mathbf{h}_{\mathcal{I}(v)}^k \leftarrow \mathrm{AGG}^k\left(\{\mathbf{h}_e^k, \forall e \in \mathcal{I}(v)\}\right), \ \forall v \in \mathcal{V}$
    $\mathbf{h}_v^k \leftarrow \sigma\left[\mathbf{W}_2^k \cdot \{\mathbf{h}_v^{k-1} \cup \mathbf{h}_{\mathcal{N}(v)}^k\} || \ \mathbf{W}_3^k \cdot \mathbf{h}_{\mathcal{I}(v)}^k\right], \forall v \in \mathcal{V}$
    $\mathbf{h}_e^k \leftarrow \sigma(\mathbf{h}_e^k), \ \forall e \in \mathcal{E}$
  else
    $\mathbf{h}_v^k \leftarrow \mathbf{W}_2^k \cdot \{\mathbf{h}_v^{k-1} \cup \mathbf{h}_{\mathcal{N}(v)}^k\}, \ \forall v \in \mathcal{V}$
    $\mathbf{h}_{\mathcal{I}(e)}^k \leftarrow \mathrm{AGG}^k\left(\{\mathbf{h}_v^k, \forall v \in \mathcal{I}(e)\}\right), \ \forall e \in \mathcal{E}$
    $\mathbf{h}_e^k \leftarrow \sigma\left[\mathbf{W}_1^k \cdot \mathbf{h}_e^{k-1} || \ \mathbf{W}_4^k \cdot \mathbf{h}_{\mathcal{I}(e)}^k\right], \ \forall e \in \mathcal{E}$
    $\mathbf{h}_v^k \leftarrow \sigma(\mathbf{h}_v^k), \ \forall v \in \mathcal{V}$
  end if
end for

▷ Feed-forward networks for node and edge predictions

$\hat{y}_v \leftarrow \mathrm{MLP}_{\mathcal{V}}(\mathbf{h}_v^K), \ \forall v \in \mathcal{V}$
$\hat{f}_e \leftarrow \mathrm{MLP}_{\mathcal{E}}(\mathbf{h}_e^K), \ \forall e \in \mathcal{E}$

The predicted node and edge probabilities can be used to train the embedding model and the classifier(s) using a joint loss approach. This joint loss is defined as

$$\mathcal{L} = \sum_{e \in \mathcal{E}} l(\hat{f}_e, f_e) + \sum_{v \in \mathcal{V}} l(\hat{y}_v, y_v), \tag{9}$$

where $l(\cdot)$ refers to the cross-entropy loss. When the target predictions have a highly skewed distribution, we apply a weighted cross-entropy loss to give more weight to scenarios from the minority class. For instance, in cascading failure scenarios where failures occur less frequently, we weigh the buses and branches with target status *fail* higher than the ones with target status *operational*.

Empirical Evaluation

Datasets

The IEEE test case systems have been widely used for benchmarking methods developed for power systems, including predictive ML tasks (Varbella, Gjorgiev, and Sansavini 2023; Liu et al. 2021; Zhu et al. 2022). In this work, we evaluate graph-based approaches using the standard 39-bus and the 118-bus test systems provided in the MATPOWER toolkit in Matlab (Zimmerman, Murillo-Sánchez, and Thomas 2011). For 118-bus that does not contain line ratings, we use values from (Blumsack 2006). We further reduce some line ratings to induce severe failures in the networks. Using AC-CFM as described in the earlier section, we generate 5000 scenarios for each of the test case systems. We run all simulations using the default parameters of AC-CFM and we use the Newton-Raphson method from MATPOWER as the power flow solver.

Dataset	39-bus	118-bus
#Buses	39	118
#Branches	46	186
#Contingencies	1:10	1:40
% Total bus failures	40.5	12.1
% Total branch failures	51.3	24.4
Bus label homophily	85.2	92.6
Branch label homophily	79.6	78.5

Table 1: Dataset statistics

Table 1 reports the statistics regarding the scenarios generated under these settings. We report the target labels' distributions by calculating an aggregate over the number of failed buses and branches in each scenario. The label homophily measure indicates how likely two connected buses (branches) are to have the same status on average over all scenarios. Using the power flow solutions of each scenario, we design meaningful bus and branch features based on the protection mechanisms triggered in the AC-CFM model when performing the simulations. For each bus, we obtain 9 computed using the bus type, real power generation, reactive power generation, voltage magnitude, voltage angle, voltage limits, and flow limits. Similarly, for branches, we obtain 9 features computed using information regarding the operational status, resistance, reactance, line ratings, active power flow, reactive power flow, and branch flow ratio.

Table 2: Performance comparison of different GNN models

Data	Model	Branch		Bus	
		Accuracy (%)	Balanced accuracy (%)	Accuracy (%)	Balanced accuracy (%)
39-bus	GCN	68.5±1.2	68.1±1.2	75.9±0.8	73.3±1.2
	GraphSAGE	80.8±0.2	80.7±0.2	**82.8±0.6**	**81.3±0.8**
	GIN	81.4±0.4	81.3±0.4	80.1±0.4	78.5±0.6
	GAT-A	74.9±1.7	74.7±1.7	78.8±0.8	76.6±1.2
	GAT-B	80.5±0.6	80.3±0.6	80.9±0.5	79.0±0.6
	NEA-GNN	**82.0±0.9**	**82.0±0.8**	80.6±1.1	79.1±1.3
118-bus	GCN	59.0±16.4	63.1±6.4	87.8±0.3	50.0±0.0
	GraphSAGE	70.9± 2.0	73.0±0.9	89.6±0.3	**64.3±1.0**
	GIN	60.8± 1.9	68.3±1.0	88.1±0.3	59.1±0.6
	GAT-A	65.0± 2.1	70.9±1.1	88.7±0.5	60.2±0.7
	GAT-B	76.2± 0.8	76.4±0.3	88.9±0.2	62.0±1.9
	NEA-GNN	**88.8± 0.5**	**79.5±1.7**	**89.7±0.3**	60.7±1.2

Experimental Setup

We describe the model architectures and hyperparameters used for training and evaluation in this section. For each dataset, we randomly split the 5000 scenarios into training, validation, and test sets with a 60-20-20 ratio respectively. To obtain node and/or edge embeddings we implement GCN, GraphSAGE, GAT, GIN, and NEA-GNN discussed in prior sections. For GAT, we train two variations, GAT-A which uses only node features to compute attention coefficients, and GAT-B which additionally uses edge features for attention. Both models, however, output only node embeddings at the final layer. The embedding models are implemented with 2 layers of convolution and 16 hidden dimensions and mean aggregation. We use a fully connected feed-forward neural network with 2 layers and 128 hidden dimensions as the classification module. Where applicable we define separate classification modules for the nodes and edges. We use ReLU as the activation function followed by dropout and train all models with Adam optimizer. We further fix the learning rate at 0.001 and the batch size at 64. For class-weighted loss, we vary the weights in $\{0.1, 0.5, 1.0\}$. We use the same parameter settings for all baseline models as well. We conduct experiments with 10 random seeds and report the performance as an average along with their standard deviation. Performance is evaluated in terms of accuracy and balanced accuracy as one of the datasets has a heavily imbalanced class distribution. All models are implemented using PyTorch-Geometric (Fey and Lenssen 2019) and are run on GPU Tesla V100 (32GB RAM). The datasets and implementation are available at https://github.com/karuna-bhaila/gnn-cascading-failure.

Results

We report the results of our experiments for all GNN models in Table 2. For the 39-bus system, we observe that the proposed method NEA-GNN outperforms all other GNN models for the branch failure prediction task with a slight trade-off in the bus failure prediction task. More notably, compared to GAT-B which uses branch features only in the attention module, NEA-GNN implements message-passing using both node and edge features which allows the model to learn better edge representations and more accurate branch prediction results. We also observe that GAT-B has a 6% gain in branch prediction accuracy compared to GAT-A which further highlights the contribution of edge features in graph convolutions albeit indirectly. Despite not using edge features, GraphSAGE and GIN show good performance. We conjecture that this is due to the physical proximity of the tripped buses and branches and the small network size.

For the 118-bus dataset, the proposed model NEA-GNN outperforms all models for both branch and bus prediction tasks in terms of accuracy. Specifically, NEA-GNN improves branch prediction performance by a fairly large margin over the baselines. Altogether, NEA-GNN has a collective improvement of approximately 8% over GAT-B when considering bus and branch prediction simultaneously. Similar to the 39-bus dataset, we observe that GAT-B improves over GAT-A's performance with an 11% gain in accuracy and additionally performs better than the other three baselines. These results demonstrate the effectiveness of utilizing branch features for graph convolutions. In particular, using edge features for message-passing in NEA-GNN allows the model to propagate the failure status of the branches in the contingency set throughout the network since the initial status is included as one of the edge features. Furthermore, considering both datasets, NEA-GNN has the least trade-off between bus and branch prediction performances. This implies that the proposed convolution approach can scale well between datasets of different sizes, especially compared to baselines such as GCN which performs quite poorly for the 118-bus dataset; the bus prediction balanced accuracy of GCN indicates that the model outputs are synonymous with majority class voting, i.e., it predicts the same majority class for all nodes. Thus, we conclude that the proposed GNN model can effectively utilize node and edge attributes to learn embeddings simultaneously with better collective performances for bus and branch failure prediction on different network sizes.

Conclusion and Future Work

In this work, we conducted an experimental study on the use of graph-based convolution approaches for end-to-end

prediction of cascading failures in power grids. First, we created a large dataset by simulating cascading failures conditioned on branch failures using an AC-based cascading failure model. We evaluated state-of-the-art GNNs on the simulated dataset for failure prediction in power grids. We also formulated and evaluated a convolution method that performs implicit message-passing between graph edges. The model uses carefully selected grid attributes and power flow variables as features and predicts the failure status of buses and branches. Our experimental results demonstrate that attributed graph convolution methods are capable of learning to predict component failure status just from observing their starting state potentially helping in identifying high-risk areas to implement precautionary measures.

There are several possible future works in this direction. Here, we evaluated our methods on scenarios that were not observed during training but generated from the same grid system nonetheless. We will extend this work to study GNN performance in the transfer learning scenario where the grid systems used to generate training and testing scenarios are different. Furthermore, in contrast to the implicit message-passing explored here, we will study the role of edge features and develop models that perform direct message-passing between them to facilitate learning in feature-rich graphs. Moreover, in this study, we assumed local failure propagation whereas, non-local and long-range failure propagation may also have disastrous consequences. We will investigate these phenomena in subsequent works.

Acknowledgements

This work was supported in part by NSF 2119691. We thank Dr. Roy McCann and Mojtaba Ahanch for their helpful discussions regarding data generation.

References

Blumsack, S. 2006. *Network topologies and transmission investment under electric-industry restructuring*. Carnegie Mellon University.

Dobson, I.; Carreras, B. A.; Lynch, V. E.; and Newman, D. E. 2007. Complex systems analysis of series of blackouts: Cascading failure, critical points, and self-organization. *Chaos: An Interdisciplinary Journal of Nonlinear Science*.

Dobson, I.; Carreras, B. A.; and Newman, D. E. 2005. A loading-dependent model of probabilistic cascading failure. *Probability in the Engineering and Informational Sciences*.

Fey, M.; and Lenssen, J. E. 2019. Fast Graph Representation Learning with PyTorch Geometric. arXiv:1903.02428.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*.

Guo, H.; Zheng, C.; Iu, H. H.-C.; and Fernando, T. 2017. A critical review of cascading failure analysis and modeling of power system. *Renewable and Sustainable Energy Reviews*.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*.

Jiang, X.; Ji, P.; and Li, S. 2019. CensNet: Convolution with Edge-Node Switching in Graph Neural Networks. In *IJCAI*.

Kearnes, S.; McCloskey, K.; Berndl, M.; Pande, V. S.; and Riley, P. 2016. Molecular graph convolutions: moving beyond fingerprints. *J. Comput. Aided Mol. Des.*

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Liao, W.; Bak-Jensen, B.; Pillai, J. R.; Wang, Y.; and Wang, Y. 2021. A review of graph neural networks and their applications in power systems. *Journal of Modern Power Systems and Clean Energy*.

Liu, Y.; Zhang, N.; Wu, D.; Botterud, A.; Yao, R.; and Kang, C. 2021. Searching for Critical Power System Cascading Failures With Graph Convolutional Network. *IEEE Trans. Control. Netw. Syst.*

Noebels, M.; Preece, R.; and Panteli, M. 2022. AC Cascading Failure Model for Resilience Analysis in Power Networks. *IEEE Systems Journal*.

Pambour, K. A.; Erdener, B. C.; Bolado-Lavin, R.; and Dijkema, G. P. 2017. SAInt–A novel quasi-dynamic model for assessing security of supply in coupled gas and electricity transmission networks. *Applied energy*.

Sanchez-Gonzalez, A.; Godwin, J.; Pfaff, T.; Ying, R.; Leskovec, J.; and Battaglia, P. 2020. Learning to simulate complex physics with graph networks. In *ICML*. PMLR.

Shuvro, R. A.; Das, P.; Hayat, M. M.; and Talukder, M. 2019. Predicting Cascading Failures in Power Grids using Machine Learning Algorithms. In *2019 North American Power Symposium (NAPS)*.

Song, J.; Cotilla-Sanchez, E.; Ghanavati, G.; and Hines, P. D. H. 2016. Dynamic Modeling of Cascading Failure in Power Systems. *IEEE Transactions on Power Systems*.

Vaiman, M.; Bell, K.; Chen, Y.; Chowdhury, B.; Dobson, I.; Hines, P.; Papic, M.; Miller, S.; and Zhang, P. 2012. Risk Assessment of Cascading Outages: Methodologies and Challenges. *IEEE Transactions on Power Systems*.

Varbella, A.; Gjorgiev, B.; and Sansavini, G. 2023. Geometric deep learning for online prediction of cascading failures in power grids. *Reliability Engineering & System Safety*.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *ICLR 2019*.

Zhang, M.; Fu, S.; Yan, J.; Zhang, H.; Ling, C.; Shen, C.; and Shi, P. 2021. An XGBoost-Based Vulnerability Analysis of Smart Grid Cascading Failures under Topology Attacks. In *2021 IEEE International Conference on Systems, Man, and Cybernetics*.

Zhu, Y.; Zhou, Y.; Wei, W.; and Wang, N. 2022. Cascading Failure Analysis Based on a Physics-Informed Graph Neural Network. *IEEE Transactions on Power Systems*.

Zimmerman, R. D.; Murillo-Sánchez, C. E.; and Thomas, R. J. 2011. MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education. *IEEE Transactions on Power Systems*.