Contingency Analysis with Warm Starter using Probabilistic Graphical Model

Shimiao Li
Carnegie Mellon University,
Pittsburgh, USA
shimiaol@andrew.cmu.edu

Amritanshu Pandey
University of Vermont
Burlington, USA
amritanshu.pandey@uvm.edu

Larry Pileggi
Carnegie Mellon University,
Pittsburgh, USA
pileggi@andrew.cmu.edu

***Abstract*—Cyberthreats are an increasingly common risk to the power grid and can thwart secure grid operations. We propose to extend contingency analysis to include cyberthreat evaluations. However, unlike the traditional N-1 or N-2 contingencies, cyberthreats (e.g., MadIoT) require simulating hard-to-solve N-k (with k $>>$ 2) contingencies in a practical amount of time. Purely physics-based power flow solvers, while being accurate, are slow and may not solve N-k contingencies in a timely manner, whereas the emerging data-driven alternatives are fast but not sufficiently generalizable, interpretable, and scalable. To address these challenges, we propose a novel conditional Gaussian Random Field-based data-driven method that performs fast and accurate evaluation of cyberthreats. It achieves speedup of contingency analysis by warm-starting simulations, i.e., improving starting points, for the physical solvers. To improve the physical interpretability and generalizability, the proposed method incorporates domain knowledge by considering the graphical nature of the grid topology. To improve scalability, the method applies physics-informed regularization that reduces model complexity. Experiments validate that simulating MadIoT-induced attacks with our warm starter becomes approximately 5x faster on a realistic 2000-bus system.**

***Index Terms*—contingency analysis, cyber attack, Gaussian random field, power flow, warm start**

I. Introduction

The power grid is becoming increasingly challenging to operate. Climate change is accelerating the frequency of natural disasters, causing a higher rate of equipment failure, and new-age grid resources such as renewables and distributed energy resources are causing more uncertainty in supply. With the growing uncertainties and threats, today's grid operators use contingency analysis (CA) to identify the grid's vulnerability against them.

CA [1] is a simulation module located within the grid control centers' Energy Management System (EMS). Grid operators (and planners) use it to perform a set of "what if" power flow simulations to preemptively evaluate the impacts of selected disturbances and outages on the power grid health indicators (e.g., voltages and line flow). If the output from CA indicates that the grid is operating outside of its operational limits, the decision-makers can maintain reliability by taking remedial action, for example, redispatching generation via a security-constrained optimal power flow (SCOPF). Today in operations, real-time CA is run every 5-30 minutes (e.g., Electric Reliability Council of Texas (ERCOT) which is the independent system operator in Texas runs it every 5 min [2] and North American Electric Reliability Corporation (NERC): requires that CA runs at least once every 30 min). Since it is computationally prohibitive to simulate all combinations of component failures in a limited time, the operators only include a predefined set of N-1 (loss of one component) contingencies in CA. These contingencies correspond to failures due to mechanical issues or natural disasters.

Today's CA methods may not suffice when facing a modern grid disturbance in the form of cyberthreat [3]. Unlike N-1 contingencies, cyber threat can cause outages and malicious changes simultaneously at numerous locations, which represents N-k contingencies with $k >> 1$. Recent years' literature has documented many cyber attacks that target multiple locations. Some describe a brute-force attack on many critical control devices (e.g. toggling main breakers of generators), causing blackout for days [4][5]; some describe hacking into a large set of grid-edge devices (e.g., Internet of Things-IoT devices) to deny or degrade electric service [6], and some describe disrupting the confidentiality and integrity of power grid data through the modification of many data values [7].

With the emergence of these cyberthreats, the operators should preemptively secure the grid against them. As such, in the future, CA should also include cyber events within its predefined set of contingencies. Simulating the power flow impact of these cyber events can inform the grid operators of the grid's susceptibility to these attacks and also filter out potential incidents where preventive actions are needed.

In this paper, we focus on a popular type of cyberthreat that can cause N-k contingencies: the MadIoT attack, also called BlackIoT attack, which has emerged due to the proliferation of the IoT devices (i.e., grid edge devices). This theoretical attack model succeeds by hacking into the high-wattage IoT-controlled load devices and adversely changing their load demand to disrupt the grid operation. Attack instances of the MadIoT threat represent N-k events due to the concurrent load manipulations at many locations. As this paper extends the functionality of contingency analysis unit to also simulate such modern cyberattacks, we implicitly extended the definition of

Accepted by the 23rd Power Systems Computation Conference (PSCC 2024).

contingency to include the manipulation (which might not be a traditional failure) of devices that might make the system unstable.

Unfortunately, the computational and analytical techniques in the EMS today may not solve CA robustly and fast enough for a large set of N-k contingencies events in an allocated amount of time, like within $\sim 5-30$ min for real-time CA. The main reason is that while fast evaluation of N-1 contingencies is possible due to the close proximity of post-contingency solution to the pre-contingency solution, these characteristics no longer hold true for N-k contingencies. Severe N-k contingencies can cause a big shift in the grid states, making it difficult to find *sufficiently good* initial conditions that can lead to *accurate* and *fast* simulation.

Learning-based techniques, which can develop high-performance function approximators, provide an efficient way to address this challenge. Learning from some existing (simulated) N-k contingency data enables approximating the mapping from an original system to its state after a given disturbance. The goal of this paper is to develop such an approximator which serves as a warm starter that provides *good and fast* initialization to CA and enables it to converge in fewer iterations. Specifically, given a pre-contingency power grid and corresponding contingency information, the warm starter predicts the post-contingency bus voltages. The prediction is then used as the initial point for simulating the contingency.

Developing a data-driven learning-based method that is practical for grid-specific applications is important. General machine learning (ML) tools for physical systems exhibit the drawbacks of requiring a large amount of training data and outputting non-physical solutions. As an improvement, many current works of grid simulation and optimization have explored the use of physics, i.e., domain knowledge or domain expertise, towards developing physics-informed machine learning (ML) approaches [8][9]-[10][11][12][13][14][15][16][17][18][19][20]. However, we have found that many methods still lack sufficient generalization, interpretability and scalability. Section II-A provides an overview of these current efforts and their limitations.

To address the limitations in the current physics-informed ML methods while developing a data-driven warm starter, this paper builds a probabilistic graphical model where the conditional joint distribution is factorized into a pairwise form. The potential functions (which are components of the factorization) are then defined in the form of Gaussian functions, giving rise to a conditional Gaussian Random Field (GRF) to model the conditional joint distribution. Neural networks are used to map from local inputs to the unknown parameters in local Gaussian potential functions, the training of which is based on a maximization of the conditional likelihood. Such an integration of GRF with neural networks aims at improved 1) model generalizability by incorporating topology changes in the grid into the method by using architecture-constrained graphical models, 2) physical interpretability, since the inference model of our Gaussian Random Field has been found to form a linear proxy of the power system, and 3) trainability and scalability by using a graphical model with physics-informed regularization techniques (e.g., parameter sharing). The results show that on a 2000-bus system, the proposed warm starter enabled contingency simulation achieves 5x faster convergence than the traditional initialization methods.

Reproducibility: our code is publicly available at https://github.com/ohCindy/GridWarm.git.

II. Related Work

A. Physics-informed ML for power grids: literature review

Many prior works have included domain-knowledge in their methods to address the problem of missing *physics* in generic ML tools. These methods collectively fall under physics-informed ML paradigm for power grid operation, control, and planning and can be broadly categorized into following categories:

1) Reducing search space: In these methods, domain knowledge is used to narrow down the search space of parameters and/or solutions. For example, [8] designed a grid topology controller which combines reinforcement learning (RL) Q-values with power grid simulation to perform a *physics*-guided action exploration, as an alternative to the traditional epsilon-greedy search strategy. Works in [9]-[10] studied the multi-agent RL-based power grid control. In these approaches the power grid is partitioned into controllable sub-regions based on domain knowledge (i.e., electrical characteristics) to reduce the high-dimensional continuous action space into lower-dimension sub-spaces which are easier to handle.

2) Enforcing system constraints and technical limits: Many recent works apply deep learning to power grid analysis problems. These include but are not limited to power flow (PF) [11][12][13], DC optimal power flow (DCOPF)[14], ACOPF [15][16][17] and state estimation (SE) [18][19][20]. These methods are generally based on (supervised) learning of an input-to-solution mapping using historical system operational data or synthetic data. A type of method amongst these works is unrolled neural networks [18][19][20] whose layers mimic the iterative updates to solve SE problems using first-order optimization methods (i.e., gradient descent methods), based on quadratic approximations of the original problem. Many methods learn the 'one-step' mapping function. Among these, some use deep neural network (DNN) architectures [11][12][15][14] to learn high-dimensional input-output mappings, some use recurrent neural nets (RNNs) [21] to capture grid dynamics, and others apply graph neural networks (GNN) [13][16][22] to capture the exact topological structure of power grid.

To promote *physical* feasibility of the solution, many works impose equality or inequality system constraints by i) encoding hard constraints inside NN layers (e.g. using sigmoid layer to encode technical limits of upper and lower bounds), ii) applying prior on the NN architecture (e.g., Hamiltonian [23] and Lagrangian neural networks [24]), iii) augmenting the objective function with penalty terms in a supervised [11] or unsupervised [15][12] way, iv) projecting outputs [14] to the feasible domain, or v) combining many different strategies. In

all these methods, incorporating (nonlinear) system constraints remains a challenge, even with state-of-the-art toolboxes [25], and most popular strategies lack rigorous guarantees of nonlinear constraint satisfaction.

While these methods have advanced the state-of-the-art in physics-informed ML for power grid applications, critical limitations in terms of generalization, interpretation, and scalability exist. We discuss these further:

Limited generalization: Many existing methods do not adapt well to changing grid conditions. Take changes in network topology as an example. Many current works are built on non-graphical architectures without any topology-related inputs. These, once trained, only work for one fixed topology and cannot generalize to dynamic grid conditions. More recently, some works have begun to encode topology information. Graph model-based methods (e.g., GNN [13][16][22]) naturally impose topology as a hard constraint and thus can account for topology changes. Alternatively, work in [12] encodes the topology information into the penalty term (as a soft constraint) through the admittance and adjacency matrix, and [20] accounts for topology in NN implicitly by applying a topology-based prior through a penalty term. While these methods lead to better topology adaptiveness, they also have some risks: the use of penalty terms [12][20] to embed topology information as a soft constraint can lead to limited precision; and, for problems (like OPF) where information needs to be exchanged between far-away graph locations, the use of GNNs requires carefully designed global context vectors to output predictions with global-level considerations.

Limited interpretability: Despite that many ML models (NN, decision trees, K-nearest-neighbors) are universal approximators, interpretations of their functionality from a physically meaningful perspective are still very limited. The general field of *model interpretability* [26] focuses on explaining *how a model works*, to mitigate fears of the unknown. Broadly, investigations of interpretability have been categorized into transparency (also called ad-hoc interpretations) and post-hoc interpretations. The former aims to elucidate the mechanism by which the ML *blackbox* works before any training begins by considering the notions of *simulatability* (Can a human work through the model from input to output, in reasonable time steps through every calculation required to produce a prediction?), *decomposability* (Can we attach an intuitive explanation to each part of a model: each input, parameter, and calculation?), and *algorithmic transparency* (Does the learning algorithm itself confer guarantees on convergence, error surface even for unseen data/problems?). And *post-hoc interpretation* aims to inspect a learned model after training by considering its natural language explanations, visualizations of learned representations/models, or explanations of empirical examples. However, none of these concepts in the field of ML model interpretability formally evaluates *how a ML model makes predictions in a physically meaningful way* when it is used on an industrial system like the power grid. Some recent works have explored the physical meaningfulness of their models from the power system perspective. Still, interpretations are made in conceptually different ways without uniform metric: Unrolled neural networks (which has been used as data-driven state estimation for power grid [20][18]) are more decomposable and interpretable in a way that the layers mimic the iterations in the physical solvers, yet these models [19][18][20] mainly unroll first-order solvers instead of the second-order (Newton-Raphson) realistic solvers. GNN-based models [13][16][22] naturally enable better interpretability in terms of representing the graph structure. Work in [11] provides some interpretation of its DNN model for PF by matching the gradients with power sensitivities and subsequently accelerating the training by pruning out unimportant gradients. [12] learns a weight matrix that can approximate the bus admittance matrix; however, with only limited precision. To summarize, due to the limited interpretability, ML models still have some opacity and blackbox-ness, when compared with the purely physics-based models (e.g., power flow equations).

Scalability issues: In the case of large-scale systems, models (like DNNs) that learn the mapping from high-dimensional input-output pairs will inevitably require larger and deeper designs of model architecture and thereafter, massive data to learn such mappings. This can affect the practical use in real-world power grid analytics.

3) Extracting meaningful features or crafting an interpretable latent space: Many works exist in this class. For example, [27][28] learned the latent representation of sensor data in a graph to capture temporal dependency [27] or spatial sensor interactions [28]. [29] applied the influence model to learn, for all edge pairs, the pairwise influence matrices, which are then used to predict line cascading outages. [30] crafted a graph similarity measure from power sensitivity factors and detected anomalies in the context of topology changes by weighing historical data based on this similarity measure.

B. MadIoT: IoT-based Power Grid Cyberthreat

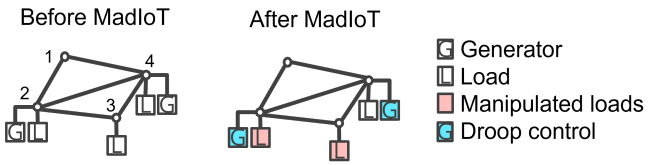


Fig. 1. A toy instance of MadIoT: a subset of loads is manipulated.

The proliferation of IoT devices has raised concerns about IoT-induced cyber attacks. [6] proposed a threat model, namely *BlackIoT* or *MadIoT*, where an attacker can manipulate the power demand by synchronously turning on/off or scaling up/down some high-wattage IoT-controlled loads on the grid. This can lead to grid instability or grid collapse. Fig. 1 illustrates an attack instance of the MadIoT threat. To evaluate the impact of a postulated attack, [6] uses a steady-state analysis of the power grid while considering the droop control, protective relaying, and thermal and voltage limits for various components.

III. A NOVEL WARM STARTER

Goal: This proposed warm starter will enable simulating a large-set of cyberthreat-driven contingencies (C_{N-k}) in a practical amount of time by improving the speed of individual simulation.

Purely physics-based solvers can give accurate solutions but are slow when they have to solve a large number of *hard-to-solve* N-k contingency events. In contrast, replacing physical solvers with purely data-driven techniques makes it fast but has severe limitations, as discussed in Section II-A. In this paper, we propose a novel physics-informed ML model that can warm-start the physical solvers when simulating *hard-to-solve* contingencies (C_{N-k}). The warm starter will predict the post-contingency voltages supplied as initial conditions to the physical solver for fast convergence. The proposed model is designed to be **generalizable to topology change** by using a graphical model, and **physically interpretable** by forming a linear system proxy, and **scalable** by application of regularization techniques on the graphical model.

A. Task definition and symbol notations

As shown in Fig. 2, given an input $\boldsymbol{x}$ which contains contingency information c and (pre-contingency) system information G, a warm starter makes prediction $\boldsymbol{y}$ which is an estimate of the post-contingency bus voltages $\boldsymbol{v}^{post}$. The model is a function mapping, which is learned from training dataset $Data = \{(\boldsymbol{x^{(j)}}, \boldsymbol{y^{(j)}})\}$, where (j) denotes the j-th sample. Table I shows the symbols used in this paper.

TABLE I
SYMBOLS AND DEFINITIONS

Symbol	Interpretation
G	case data before contingency containing topology, generation, and load settings
$\boldsymbol{v_i}$	the voltage at bus i, $\boldsymbol{v_i} = [v_i^{real}, v_i^{imag}]^T$
$\boldsymbol{v}^{pre/post}$	$\boldsymbol{v}^{pre/post} = [\boldsymbol{v}_1^{pre/post}, \boldsymbol{v}_2^{pre/post}, ..., \boldsymbol{v}_n^{pre/post}]^T$ the pre/post-contingency voltages at all buses
c	contingency setting *(type, location, parameter)* e.g. $(\textit{MadIoT}, [1,3], 150\%)$: increasing loads at bus 1 and 3 to 150% of the original value via MadIoT attack.
i, n	bus/node index; total number of nodes
(s,t)	a branch/edge connecting node s and node t
$\mathcal{V}, \mathcal{E}$	set of all nodes and edges: $i \in \mathcal{V}, \forall i; (s,t) \in \mathcal{E}$
j, N	data sample index; total number of (training) samples
$(\boldsymbol{x}, \boldsymbol{y})$	a sample with feature $\boldsymbol{x}$ and output $\boldsymbol{y}$ $\boldsymbol{y} = [\boldsymbol{y_1}, ..., \boldsymbol{y_n}]^T = [\boldsymbol{v}_1^{post}, ..., \boldsymbol{v}_n^{post}]^T$

B. Method Overview

Power grid can be naturally represented as a graph, as shown in Fig. 2. Nodes and edges on the graph correspond to power grid buses and branches (lines and transformers), respectively. Each node represents a variable $\boldsymbol{y_i}$ which denotes voltage phasor at bus i, whereas each edge represents a direct inter-dependency between adjacent nodes. Such an undirected graphical model that models the dependency relationship among the random variables is called a Markov Random Field (MRF).

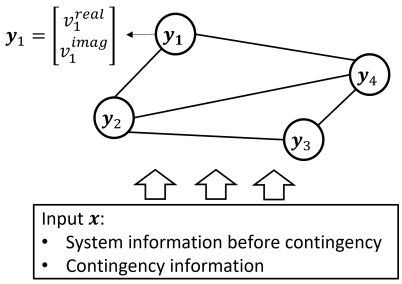


Fig. 2. A power grid can be naturally represented as a graphical model. Each node represents the bus voltage after contingency, each edge represents a branch status after contingency. Now conditioned on an original power grid G and a contingency c that happens on it, we want to know the bus voltages after contingency.

The use of MRF enables a compact way of writing the conditional joint distribution and performing inference thereafter, using observed data. Specifically, when contingency happens, the joint distribution of variables $\boldsymbol{y}$ conditioned on input features $\boldsymbol{x}$ can be further factorized in a pairwise manner, leading to a pairwise Markov Random Field [31], as Fig.1 shows:

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta}, \boldsymbol{x})} \prod_{i=1}^{n} \psi_i(\boldsymbol{y_i}) \prod_{(s,t)\in \boldsymbol{E}} \psi_{st}(\boldsymbol{y_s}, \boldsymbol{y_t}) \quad (1)$$

where $\boldsymbol{\theta}$ denotes model parameters that maps $\boldsymbol{x}$ to $\boldsymbol{y}$; $\psi_i(\boldsymbol{y_i}), \psi_{st}(\boldsymbol{y_s}, \boldsymbol{y_t})$ are node and edge potentials conditioned on $\boldsymbol{\theta}$ and $\boldsymbol{x}$; and $Z(\boldsymbol{\theta}, \boldsymbol{x})$ is called the partition function that normalizes the probability values such that they sum to the value of 1.

The factorization model in (1) is inspired by pairwise continuous MRF [31] and has an intuitive form: every edge potential encodes the mutual correlation between two adjacent nodes; both node and edge potentials represent the local contributions of nodes/edges to the joint distribution. In the task of contingency analysis, each potential function intuitively represents how the status of each bus and branch 'independently' impacts the bus voltages.

Given a training dataset of N samples $\{(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})\}$, the training and inference can be described briefly as:

- **Training:** With proper definition of the potential functions $\psi_i(\boldsymbol{y_i}|\boldsymbol{x}, \boldsymbol{\theta}), \psi_{st}(\boldsymbol{y_s}, \boldsymbol{y_t}|\boldsymbol{x}, \boldsymbol{\theta})$ (see Section III-C and III-D), and the parameter $\boldsymbol{\theta}$ can be learned by maximizing log-likelihood

$$\hat{\boldsymbol{\theta}} = arg\max_{\boldsymbol{\theta}} \sum_{j=1}^{N} logl(\theta)^{(j)} \quad (2)$$

where $l(\theta)^{(j)}$ denotes the likelihood of the j-th sample:

$$l(\theta)^{(j)} = p(\boldsymbol{y}^{(j)}|\boldsymbol{x}^{(j)}, \boldsymbol{\theta}) \quad (3)$$

- **Inference:** For any new input $\boldsymbol{x}$, we make use of the estimated parameter $\hat{\boldsymbol{\theta}}$ to make a single-point prediction

$$\hat{\boldsymbol{y}}_{\boldsymbol{test}} = arg\max_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) \tag{4}$$

The use of a probabilistic graphical setting naturally integrates the domain knowledge from grid topology into the method:

Domain knowledge of grid topology: *Power flow result is conditioned on the grid topology. Bus voltages of two adjacent buses connected and directed by a physical linkage (line or transformer) have direct interactions.*

Each sample in this method can have its topology and each output is conditioned on its input topology. The following sections will discuss how the graphical model and domain knowledge enable an efficient and physically interpretable model design.

C. Pairwise Conditional Gaussian Random Field

Upon representing the power grid and its contingency as a conditional pairwise MRF factorized in the form of (1), we need to define the potential functions $\psi_i(\boldsymbol{y_i}), \psi_{st}(\boldsymbol{y_s}, \boldsymbol{y_t})$.

This paper builds a Gaussian random field which equivalently assumes that the output variable $(\boldsymbol{y})$ satisfies multivariate Gaussian distribution, i.e., $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$ is Gaussian. The justification and corresponding benefits of using Gaussian Random Field are:

- partition function $Z(\boldsymbol{\theta}, \boldsymbol{x})$ is easier to compute due to nice properties of Gaussian distribution. Specifically in the case of Gaussian, the normalization constraint can be computed easily by calculating matrix determinant $|\Lambda|$, whereas the use of other potential functions might lead computation difficulties, potentially NP-hard [31].
- high physical interpretability due to a physically meaningful inference model. We will discuss this later.

The potential functions for Gaussian random field [31] are defined as follows:

$$\psi_i(\boldsymbol{y_i}) = exp(-\frac{1}{2}\boldsymbol{y_i}^T\boldsymbol{\Lambda_i}\boldsymbol{y_i} + \boldsymbol{\eta_i}^T\boldsymbol{y_i}) \tag{5}$$

$$\psi_{st}(\boldsymbol{y_s}, \boldsymbol{y_t}) = exp(-\frac{1}{2}\boldsymbol{y_s}^T\boldsymbol{\Lambda_{st}}\boldsymbol{y_t}) \tag{6}$$

where $\boldsymbol{\Lambda_i}$ is a matrix and $\boldsymbol{\Lambda_{st}}$ is a vector. By plugging (5) and (6) into (1), we have:

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) \propto exp(\boldsymbol{\eta}^T\boldsymbol{y} - \frac{1}{2}\boldsymbol{y}^T\Lambda\boldsymbol{y}) \tag{7}$$

where $\boldsymbol{\Lambda_i}$ and $\boldsymbol{\Lambda_{st}}$ parameters are the building blocks of matrix $\boldsymbol{\Lambda}$, and $\boldsymbol{\eta}$ is a column vector composed of all $\boldsymbol{\eta_i}$. To further illustrate, consider a post-contingency grid structure in Fig. 2. The $\boldsymbol{\eta}$ and $\boldsymbol{\Lambda}$ variables for this grid structure will be shown later (in Fig. 4 where the $\mathbf{0}$ blocks in $\boldsymbol{\Lambda}$ matrix are structural zeros representing no *active* edges at the corresponding locations).

In the model (7), both $\boldsymbol{\eta}$ and $\boldsymbol{\Lambda}$ are functions of $\boldsymbol{x}, \boldsymbol{\theta}$, i.e.,

$$\boldsymbol{\eta} = f_\eta(\boldsymbol{x}, \boldsymbol{\theta_\eta}), \boldsymbol{\Lambda} = f_\Lambda(\boldsymbol{x}, \boldsymbol{\theta_\Lambda}) \tag{8}$$

and $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$ takes an equivalent form of a multivariate Gaussian distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ ($\boldsymbol{\mu}$ is the mean and $\boldsymbol{\Sigma}$ is the covariance matrix) with

$$\boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}, \boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} \tag{9}$$

Now based on these defined models, we seek to learn the parameter $\boldsymbol{\theta}$ through maximum likelihood estimation (MLE). The log-likelihood of each data sample can be calculated by:

$$l(\boldsymbol{\theta}) = logP(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^T\boldsymbol{\Lambda}\boldsymbol{y} + \boldsymbol{\eta}^T\boldsymbol{y} - logZ(\boldsymbol{\theta}, \boldsymbol{x}) \tag{10}$$

and the MLE can be written equivalently as an optimization problem that minimizes the negative log-likelihood loss on the data set of N training samples:

$$\min_{\boldsymbol{\theta}} -\sum_{j=1}^{N} l(\boldsymbol{\theta})^{(j)} \tag{11}$$

Inference and Interpretation: Upon obtaining the solution of $\hat{\boldsymbol{\theta}} = [\hat{\boldsymbol{\theta}_\eta}, \hat{\boldsymbol{\theta}_\Lambda}]^T$, parameters $\hat{\boldsymbol{\Lambda}} = f_\Lambda(\boldsymbol{x_{test}}, \hat{\boldsymbol{\theta}})$, $\hat{\boldsymbol{\eta}} = f_\eta(\boldsymbol{x_{test}}, \hat{\boldsymbol{\theta}})$ can be estimated thereafter. Then for any test contingency sample $\boldsymbol{x_{test}}$, the inference model in (4) is equivalent to solving $\hat{\boldsymbol{y}}_{\boldsymbol{test}}$ by:

$$\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{y}}_{\boldsymbol{test}} = \hat{\boldsymbol{\eta}} \tag{12}$$

Notably, the model in (12) can be seen as a **linear system proxy** of the post-contingency grid, providing a physical interpretation of the method. $\boldsymbol{\Lambda}$ is a sparse matrix with a structure similar to the bus admittance matrix where the zero entries are 'structural zeros' representing no branch connecting buses. $\boldsymbol{\eta}$ behaves like the net injection to the network.

D. NN-node and NN-edge

Finally to implement the model, we need to specify the functions of $f_\eta(\boldsymbol{x}, \boldsymbol{\theta_\eta}), f_\Lambda(\boldsymbol{x}, \boldsymbol{\theta_\Lambda})$. Taking advantage of the sparsity of $\boldsymbol{\Lambda}$, the task here is to learn a function mapping from input $\boldsymbol{x}$ to only some edge-wise parameters $\boldsymbol{\Lambda_{st}}$ and node-wise parameters $\boldsymbol{\Lambda_i}, \boldsymbol{\eta_i}$. Yet the number of $\boldsymbol{\Lambda_i}, \boldsymbol{\Lambda_{st}}, \boldsymbol{\eta_i}$ parameters still increases with grid size, meaning that the input and output size of the model will explode for a large-scale system, requiring a much more complicated model to learn a high-dimensional input-output map.

To efficiently reduce the model size, this paper implements the mapping functions using local Neural Networks: each node has a *NN-node* to predict $\boldsymbol{\Lambda_i}, \boldsymbol{\eta_i}$ using local inputs; each edge has a *NN-edge* to output $\boldsymbol{\Lambda_{st}}$ in a similar way, as shown in Fig. 3. This is inspired by our interpretation that $\boldsymbol{\Lambda}$ is a proxy of the bus admittance matrix whose elements represent some local system characteristics regarding each node and edge.

Meanwhile, to effectively learn the mapping, we must answer the following question: **how to select the input features to the NN models optimally?** We apply *domain knowledge* to design the input space that feeds most relevant features into the model:

Domain knowledge of decisive features: *The impact of contingency depends heavily on the importance of contingency*

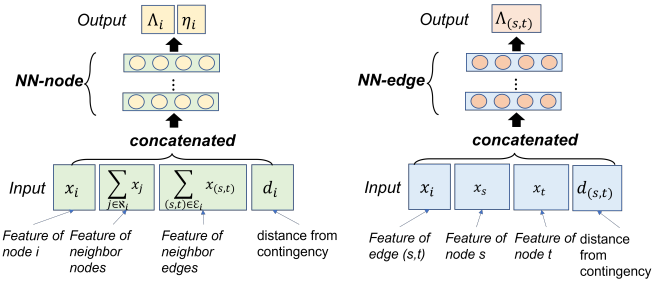


Fig. 3. each node has a *NN-node* and each edge has a *NN-edge*, to map the input features to the post-contingency system characteristics.

components which can be quantified by the amount of its generation, load or power delivery.

Domain knowledge of Taylor Expansion on system physics: *Let $v = h(G)$ denote any power flow simulation that maps the case information to the voltage profile solution. By Taylor Expansion, the post-contingency voltage can be expressed as a function depending on pre-contingency system G_{pre} and the system change ΔG caused by contingency:*

$$\boldsymbol{v^{post}} = h(G_{pre}) + h'(G_{pre})\Delta G + \frac{1}{2}h''(G_{pre})\Delta G^2 + ...$$

Therefore, the key features of the pre-contingency system (G_{pre}) and system change (ΔG) are selected as node features to feed into the NN mappings, and include:

- node feature $\boldsymbol{x}_i$: real and imaginary voltages (v_i^{real}, v_i^{imag}), power and current injections $(P_i, Q_i, I_i^{real}, I_i^{imag})$, and shunt injections $(Q_{shunt,i})$ before a contingency, and change in power injections $(\Delta P_{gen,i}, \Delta P_{load,i}, \Delta Q_{load,i})$ post contingency
- edge feature $\boldsymbol{x}_{s,t}$: line admittance and shunt parameters of the transmission line, $\boldsymbol{x}_{s,t} = [G, B, B_{sh}]$

E. Training the model with a surrogate loss

With the conditional GRF model defined in Section III-C and the NN models designed in Section III-D, the training process is illustrated in Fig. 4, where the forward pass of NN-node and NN-edge gives $\boldsymbol{\Lambda}, \boldsymbol{\eta}$, and then the loss defined from the cGRF can be calculated to further enable a backward pass that updates the parameter $\boldsymbol{\theta}$.

As described in (10)-(11), the loss function is the negative log-likelihood loss over the training data. Making use of the nice properties of Gaussian distribution, the partition function $Z(\boldsymbol{x}, \boldsymbol{\theta})$ in the loss can be calculated analytically:

$$\begin{aligned} Z &= \int_{\boldsymbol{y}} exp(\boldsymbol{\eta}^T\boldsymbol{y} - \frac{1}{2}\boldsymbol{y}^T\Lambda\boldsymbol{y})d\boldsymbol{y} \\ &= \sqrt{\frac{2\pi}{|\Lambda|}}exp(\frac{\mu^T\Lambda\mu}{2}) = \sqrt{\frac{2\pi}{|\Lambda|}}exp(\frac{\eta^T\Lambda^{-1}\eta}{2}) \end{aligned} \tag{13}$$

The detailed derivation is documented in Appendix VII-A.

Furthermore, to enable a valid distribution $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$ and unique solution during inference, it is required that the $\boldsymbol{\Lambda}$ matrix is positive definite (PD), i.e., $\boldsymbol{\Lambda} \succ \boldsymbol{0}$. Therefore, adding this constraint and substituting (13) into the loss function, the

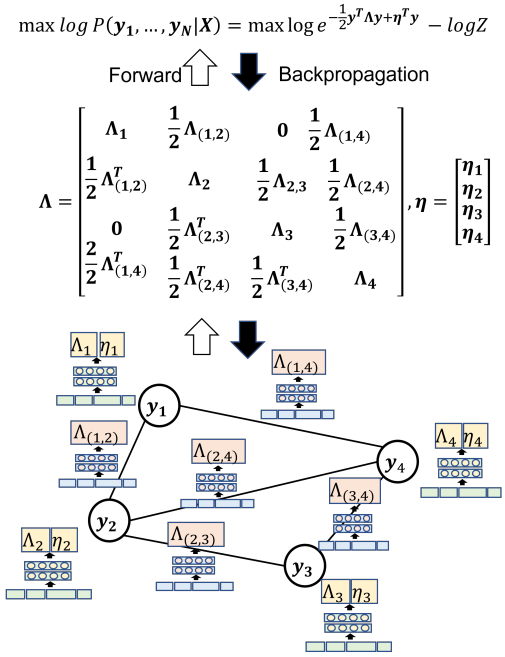


Fig. 4. Training of the proposed method: forward pass and back-propagation.

optimization problem of the proposed method can be written as:

$$\begin{aligned} \min_{\boldsymbol{\theta}} & \sum_{j=1}^{N} \frac{1}{2}\boldsymbol{y}^{(j)T}\boldsymbol{\Lambda}^{(j)}\boldsymbol{y}^{(j)} \\ & - \boldsymbol{\eta}^{(j)T}\boldsymbol{y}^{(j)} - \frac{1}{2}log|\boldsymbol{\Lambda}^{(j)}| + \frac{1}{2}\boldsymbol{\eta}^{(j)T}\boldsymbol{\Lambda}^{-1(j)}\boldsymbol{\eta}^{(j)} \end{aligned} \tag{14}$$

$s.t.$

$$\text{(forward pass) } \boldsymbol{\Lambda}^{(j)} = f_\Lambda(\boldsymbol{x}^{(j)}, \boldsymbol{\theta}_{\boldsymbol{\Lambda}}), \forall j \tag{15}$$

$$\text{(forward pass) } \boldsymbol{\eta}^{(j)} = f_\eta(\boldsymbol{x}^{(j)}, \boldsymbol{\theta}_{\boldsymbol{\eta}}), \forall j \tag{16}$$

$$\text{(positive definiteness) } \boldsymbol{\Lambda}^{(j)} \succ \boldsymbol{0}, \forall j \tag{17}$$

In this problem, maintaining the positive definiteness of matrix $\boldsymbol{\Lambda}$ for every sample is required, not only in the final solution (i.e., throughout the training process due to the $log|\boldsymbol{\Lambda}|$ in the loss). This can be computationally challenging, especially when the network size grows with large matrix dimensions.

To address this issue, we design a **surrogate loss** $\sum_{j=1}^{N}\frac{1}{2}(\boldsymbol{y}^{(j)} - \boldsymbol{\mu}^{(j)})^T(\boldsymbol{y}^{(j)} - \boldsymbol{\mu}^{(j)})$, which acts as a proxy for the actual loss we want to minimize. With the use of surrogate loss function, the overall problem converts into the following form:

$$\min_{\boldsymbol{\theta}} \sum_{j=1}^{N} \frac{1}{2}(\boldsymbol{y}^{(j)} - \boldsymbol{\mu}^{(j)})^T(\boldsymbol{y}^{(j)} - \boldsymbol{\mu}^{(j)}) \tag{18}$$

$s.t.$

$$\text{(forward pass) } \boldsymbol{\Lambda}^{(j)} = f_\Lambda(\boldsymbol{x}^{(j)}, \boldsymbol{\theta}_{\boldsymbol{\Lambda}}), \forall j \tag{19}$$

$$\text{(forward pass) } \boldsymbol{\eta}^{(j)} = f_\eta(\boldsymbol{x}^{(j)}, \boldsymbol{\theta}_{\boldsymbol{\eta}}), \forall j \tag{20}$$

$$\text{(inference) } \boldsymbol{\mu}^{(j)} = \boldsymbol{\Lambda}^{-1(j)}\boldsymbol{\eta}^{(j)}, \forall j \tag{21}$$

Appendix VII-B shows how the new loss surrogate mathematically approximates the original objective function. In this way, we removed the need to maintain positive-definiteness of the $\mathbf{\Lambda}$ in the learning process, whereas the prediction is made using a $\mathbf{\Lambda}$ computed from the forward pass, and thus it still considers the power grid structure enforced by the graphical model.

From decision theory, both the original and the surrogate loss aim to return an optimized model whose prediction $\hat{\boldsymbol{y}}$ approximates the ground truth $\boldsymbol{y}$, and both make predictions by finding out the linear approximation of the post-contingency system $\hat{\mathbf{\Lambda}}\hat{\boldsymbol{y}} = \hat{\boldsymbol{\eta}}$.

Additionally, this surrogate optimization model can be considered as minimizing the mean squared error (MSE) loss $\frac{1}{2}||\boldsymbol{y} - \hat{\boldsymbol{y}}||^2$ over the training data, where $\hat{\boldsymbol{y}}$ is the prediction (inference) made after a forward pass.

IV. Incorporating More Physics

A. Parameter sharing: a powerful regularizer

With each node having its own *NN-node* and each edge having its *NN-edge*, the number of parameters grows approximately linearly with grid size (more specifically, the number of nodes and edges). Can we reduce the model size further? The answer is yes! One option is to make all nodes share the same *NN-node* and all edges share the same *NN-edge*, so there are only two NNs in total.

Why does this work? Such sharing of *NN-node* and *NN-edge* is an extensive use of **parameter sharing** to incorporate domain knowledge into the network. Especially, from the physical perspective:

Domain knowledge of location-invariant (LI) properties: *the power grid and the impact of its contingencies have properties that are invariant to change of locations: 1) any location far enough from the contingency location will experience little local change. 2) change in any location will be governed by the same mechanism, i.e., the system equations and Kirchhoff's laws.*

Parameter sharing across the grid network significantly lowers the number of unique model parameters. Also, it reduces the need for a large amount of training data to adequately learn the system mapping for larger grid sizes (like the networks representing continental U.S. networks with $> 80k$ nodes).

B. Zero-injection bus

Domain knowledge of zero-injection (ZI) buses: *A bus with no connected generation or load is called a zero-injection (ZI) bus. These buses neither consume nor produce power, and thus, injections at these buses are zero.*

In the proposed approach, the model parameter η serves as a proxy to bus injections; therefore, we can integrate domain knowledge about zero-injection nodes into the method by setting $\boldsymbol{\eta}_i = 0$ at any ZI node i.

V. Experiments

This section runs experiments for CA instances in the context of MadIoT attack. The experiments are to validate that the proposed warm starter provides good initial states for *hard-to-solve* N-k contingencies and enables faster convergence when compared to traditional initialization techniques.

We test three versions of the proposed method. These versions differ in the level of domain-knowledge that they incorporate within their model. Table II summarizes the domain-knowledge in these versions. Table III categorizes the domain knowledge in each version.

TABLE II
Summary of domain knowledge

Knowledge	Technique	Benefits
topology	graphical model	- physical interpretability - generalization (to topology)
decisive features	feature selection	- accuracy - physical interpretability - generalization (to load&gen)
taylor expansion	feature selection	- accuracy - physical interpretability
LI properties	parameter sharing (PS)	- trainability, scalability - generalization (↓ overfitting)
ZI bus	enforce $\eta_i = 0$	- physical interpretability - generalization

TABLE III
Categorization of domain-knowledge in the 3 versions

Knowledge & techniques	cGRF	cGRF-PS	cGRF-PS-ZI
graphical model (cGRF)	✓	✓	✓
feature selection	✓	✓	✓
parameter sharing (PS)		✓	✓
ZI buses			✓

A. Data generation and experiment settings

We generate synthetic MadIoT contingencies for the following two networks: i) IEEE 118 bus network [32] ii) synthetic Texas ACTIVSg2000 network [33]. For each network, we generate N_{data} contingency samples $\{(x^{(j)}, y^{(j)})\}_{N_{data}}$ where feature the notation of x and y has been illustrated in Fig. 2 and Table I. The algorithm to generate the synthetic contingency data is given in Algorithm 1.

Contingency set and model generalization: In our experiment, we train and test warm starter for a MadIoT scenario of increasing the top K largest loads by the same amount (percentage) which is a severe scenario threatening the power grid. And the pre-contingency load is randomly sampled within the range of 95%-105% base load, and topology is randonly sampled by disconnecting 1-2 random lines on the

Algorithm 1: 3-Step Data Generation Process

Input: Base case G_{base}, type of contingency t_c, number of data samples N_{data}

Output: Generated dataset $\{(x^{(j)}, y^{(j)})\}_{N_{data}}$

for $j \leftarrow 1$ *to* N_{data} **do**

- **1. Create a random feasible pre-contingency case $G^{(j)}_{pre}$:** each sample has random topology, generation and load level.
- **2. Create contingency $c^{(j)}$ on $G^{(j)}_{pre}$:** which has attributes *type, location, parameter*.
- **3. Simulate with droop control:** run power flow to obtain the post-contingency voltages $\boldsymbol{v}^{post}$

base case, to represent the different normal operating conditions of a power system. Such contingency generation settings to a very specific setup of the dataset so that learning becomes more targeted: these contingencies are "hard-to-solve" for an optimization solver, but a simple and interpretable learned model might be able to easily extract the major relationships to provide good warm-start values. Obviously, this leads to limited generalization issue that the trained model can hardly apply to a different contingency scenario where loads are manipulated on other locations and by another amount. But in practice, this can be addressed with multiple models. The operators or decision makers can decide several other significant contingency settings that are worth consideration and evaluation. And they can train a second model on a second dataset which describes another important contingency setting. So that the different severe cyberthreat scenarios can be considered with learning performed in a targeted way.

The experiment settings that are used for the data generation, model design, and model training are documented in Table IV. Based on the idea of *NN-node* and *NN-edge* in Section III-D, the neural networks used in this work aim to learn a low-dimensional mapping from local node / edge input features to local outputs to form Λ, η. This can be effectively done with simple and shallow neural network architectures. In our experiment, the model is designed with a shallow 3-layer NN architecture with 64 hidden layers in each, to save computation time and reduce overfitting. It also allows us to experiment on whether a simple model design can give good performance. The training is then done with an Adam optimizer and step learning rate scheduler.

B. Physical Interpretability

Fig. 5 validates our hypothesis in Section III-C about *the physical interpretation of our method as a linear system proxy*. In Fig. 5, we visualize the result on a test sample to show the similarity between the linear proxy given by model parameters Λ, η and the true post-contingency system linearized admittance matrix Y_{bus} and injection current vector J at the solution; thus, validating that the model acts as a linear proxy for the post-contingency operating condition.

TABLE IV
EXPERIMENT SETTINGS

Settings	(see Table I for definitions)
N_{data}	case118: 1,000; ACTIVSg2000: 5,000 split into train, val, test set by $8:1:1$
NN-node & *NN-edge*	shallow cylinder architecture $(n\_layer, hidden\_dim) = (3, 64)$
contingency c	$type$: MadIoT $location$: randomly sampled 50% loads $parameter$: case118 200%, ACTIVSg2000 120%
optimizer	$Adam, lr = 0.001, scheduler = stepLR$

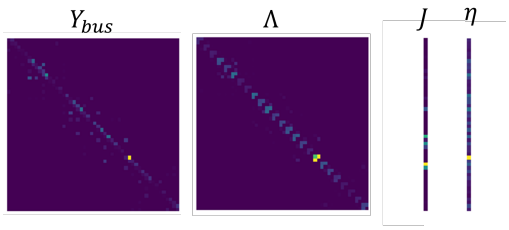


Fig. 5. Physical interpretation: This figure visualizes the values in matrices and vectors (the more yellow, the larger the value). Learned model parameters Λ, η have some similarity with the true post-contingency system admittance matrix Y_{bus} and injection current vector J, in terms of the sparse structure and value distribution. This is because the learned parameters Λ, η aims to form a linear model $\Lambda y = \eta$ which is a linear proxy of the true linearized system model $Y_{bus} y = J$.

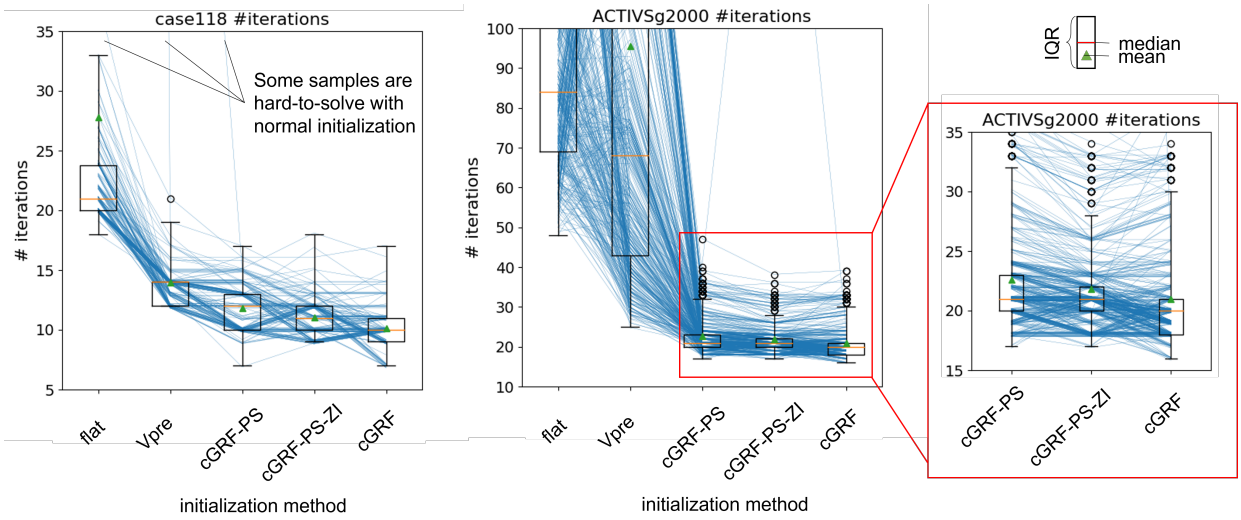
C. Application-level practicality

To verify the effectiveness of the warm starter, we compare the convergence speed (# iterations) with three different initialization methods for the physical solver [34]:

1) flat start (flat)
2) pre-contingency solution (V_{pre})
3) physical solver warm-started by the three versions of the proposed method (cGRF)

Fig. 6 shows the evaluation results on test data. These include the test samples (we split the train, validate, and test set by 8:1:1 as illustrated in Table IV) which include 100 unseen contingency samples for case 118, and 500 unseen contingency samples for ACTIVSg2000. For cGRF results, we feed the ML predictions into a power flow simulator: SUGAR [34]. The simulator is robust because it always converges. In case the general NR loop fails, the simulator uses homotopy [34] to ensure convergence, albeit at a computational cost.

The results show that simulation takes fewer iterations to converge with the proposed ML-based cGRF initialization when compared to traditional initialization methods (flat or V_{pre}). In particular, on ACTIVSg2000, many contingency samples are hard-to-solve with traditional initialization (and may require the homotopy option in SUGAR). In contrast, our ML-based cGRF method significantly speeds up convergence (up to 5x improvement in speed) even with the shallow 3-layer NN architecture.



(a) 118-bus case. (b) 2000-bus case.

Fig. 6. Result on test data: power flow simulation takes fewer iterations to converge with the proposed method, than traditional initialization methods: i. **flat** start: starting with $(V_i, \delta_i) \leftarrow (1, 0), \forall i \in \{1, ..., n\}$; ii). **Vpre** start: warm-starting from pre-contingency voltages.

Moreover, due to the use of parameter sharing (PS) which enables NNs to share parameters, the *lightweight* model cGRF-PS significantly reduces the total number of model parameters but achieves comparable results to the base model cGRF. In particular, while the average doesn't decrease significantly, the variation decreases due to the use of zero injection (ZI) knowledge. And cGRF-PS-ZI further shows that integrating more grid physics into the model through zero injection (ZI) knowledge can further improve convergence of the lightweight model.

VI. Conclusion

This paper proposes a novel physics-informed ML-based warm starter for cyberthreat-focused contingency analysis. Our method has the following features:

- **generalizability to topology changes** by using a graphical model to naturally represent grid structure
- **physically interpretability** by generating 'global' predictions from a system-level linear proxy
- **scalability** by using the graphical model and parameter sharing techniques

While being generic in its approach, the method is designed to speed up the simulation of $N - k$ contingency events. We believe these contingencies will be included in the future to evaluate the grid's vulnerability to cyberthreat instances, such as those from the MadIoT attack. In the results, we show that the proposed method can reduce the simulation iteration count by up to 5x, when compared with traditional initialization methods for such contingencies.

Acknowledgment

Work in this paper is supported in part by C3.ai Inc. and Microsoft Corporation.

References

[1] V. J. Mishra and M. D. Khardenvis, "Contingency analysis of power system," in *2012 IEEE Students' Conference on Electrical, Electronics and Computer Science*. IEEE, 2012, pp. 1–4.

[2] X. Li, P. Balasubramanian, M. Sahraei-Ardakani, M. Abdi-Khorsand, K. W. Hedman, and R. Podmore, "Real-time contingency analysis with corrective transmission switching-part i: methodology," *arXiv preprint arXiv:1604.05570*, 2016.

[3] B. Singer, A. Pandey, S. Li, L. Bauer, C. Miller, L. Pileggi, and V. Sekar, "Shedding light on inconsistencies in grid cybersecurity: Disconnects and recommendations," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2022, pp. 554–571.

[4] D. U. Case, "Analysis of the cyber attack on the ukrainian power grid," *Electricity Information Sharing and Analysis Center (E-ISAC)*, vol. 388, 2016.

[5] R. M. Lee, M. Assante, and T. Conway, "Crashoverride: Analysis of the threat to electric grid operations," *Dragos Inc., March*, 2017.

[6] S. Soltan, P. Mittal, and H. V. Poor, "Blackiot: Iot botnet of high wattage devices can disrupt the power grid," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 15–32.

[7] G. Liang, J. Zhao, F. Luo, S. R. Weller, and Z. Y. Dong, "A review of false data injection attacks against modern power systems," *IEEE Transactions on Smart Grid*, vol. 8, no. 4, pp. 1630–1638, 2016.

[8] T. Lan, J. Duan, B. Zhang, D. Shi, Z. Wang, R. Diao, and X. Zhang, "Ai-based autonomous line flow control via topology adjustment for maximizing time-series atcs," in *2020 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 2020, pp. 1–5.

[9] S. Wang, J. Duan, D. Shi, C. Xu, H. Li, R. Diao, and Z. Wang, "A data-driven multi-agent autonomous voltage control framework using deep reinforcement learning," *IEEE Transactions on Power Systems*, vol. 35, no. 6, pp. 4644–4654, 2020.

[10] M. Kamruzzaman, J. Duan, D. Shi, and M. Benidris, "A deep reinforcement learning-based multi-agent framework to enhance power system resilience using shunt resources," *IEEE Transactions on Power Systems*, vol. 36, no. 6, pp. 5525–5536, 2021.

[11] Y. Yang, Z. Yang, J. Yu, B. Zhang, Y. Zhang, and H. Yu, "Fast calculation of probabilistic power flow: A model-based deep learning approach," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2235–2244, 2019.

[12] X. Hu, H. Hu, S. Verma, and Z.-L. Zhang, "Physics-guided deep neural networks for power flow analysis," *IEEE Transactions on Power Systems*, vol. 36, no. 3, pp. 2082–2092, 2020.

[13] B. Donon, B. Donnot, I. Guyon, and A. Marot, "Graph neural solver for power systems," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.

[14] X. Pan, T. Zhao, and M. Chen, "Deepopf: Deep neural network for dc optimal power flow," in *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2019, pp. 1–6.
[15] P. L. Donti, D. Rolnick, and J. Z. Kolter, "Dc3: A learning method for optimization with hard constraints," *arXiv preprint arXiv:2104.12225*, 2021.
[16] D. Owerko, F. Gama, and A. Ribeiro, "Optimal power flow using graph neural networks," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 5930–5934.
[17] F. Diehl, "Warm-starting ac optimal power flow with graph neural networks," in *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019, pp. 1–6.
[18] L. Zhang, G. Wang, and G. B. Giannakis, "Real-time power system state estimation via deep unrolled neural networks," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2018, pp. 907–911.
[19] ——, "Real-time power system state estimation and forecasting via deep unrolled neural networks," *IEEE Transactions on Signal Processing*, vol. 67, no. 15, pp. 4069–4077, 2019.
[20] Q. Yang, A. Sadeghi, G. Wang, G. B. Giannakis, and J. Sun, "Gauss-newton unrolled neural networks and data-driven priors for regularized psse with robustness," *arXiv preprint arXiv:2003.01667*, 2020.
[21] N. Yadaiah and G. Sowmya, "Neural network based state estimation of dynamical systems," in *The 2006 IEEE international joint conference on neural network proceedings*. IEEE, 2006, pp. 1042–1049.
[22] O. Kundacina, M. Cosovic, and D. Vukobratovic, "State estimation in electric power systems leveraging graph neural networks," *arXiv preprint arXiv:2201.04056*, 2022.
[23] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
[24] M. Lutter, C. Ritter, and J. Peters, "Deep lagrangian networks: Using physics as model prior for deep learning," *arXiv preprint arXiv:1907.04490*, 2019.
[25] A. Tuor, J. Drgona, and M. Skomski, "NeuroMANCER: Neural Modules with Adaptive Nonlinear Constraints and Efficient Regularizations," 2022. [Online]. Available: https://github.com/pnnl/neuromancer
[26] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery." *Queue*, vol. 16, no. 3, pp. 31–57, 2018.
[27] Y. Yuan, Y. Guo, K. Dehghanpour, Z. Wang, and Y. Wang, "Learning-based real-time event identification using rich real pmu data," *IEEE Transactions on Power Systems*, vol. 36, no. 6, pp. 5044–5055, 2021.
[28] Y. Yuan, Z. Wang, and Y. Wang, "Learning latent interactions for event identification via graph neural networks and pmu data," *arXiv preprint arXiv:2010.01616*, 2020.
[29] X. Wu, D. Wu, and E. Modiano, "An influence model approach to failure cascade prediction in large scale power systems," in *2020 American Control Conference (ACC)*, 2020, pp. 4981–4988.
[30] S. Li, A. Pandey, B. Hooi, C. Faloutsos, and L. Pileggi, "Dynamic graph-based anomaly detection in the electrical grid," *IEEE Transactions on Power Systems*, 2021.
[31] K. P. Murphy, "Undirected graphical models (markov random fields)," *Machine Learning: A Probabilistic Perspective; MIT Press: Cambridge, MA, USA*, pp. 661–705, 2012.
[32] "Ieee 118-bus system." [Online]. Available: https://icseg.iti.illinois.edu/ieee-118-bus-system/
[33] "Activsg2000: 2000-bus synthetic grid on footprint of texas." [Online]. Available: https://electricgrids.engr.tamu.edu/electric-grid-test-cases/activsg2000/
[34] A. Pandey, M. Jereminov, M. R. Wagner, D. M. Bromberg, G. Hug, and L. Pileggi, "Robust power flow and three-phase power flow analyses," *IEEE Transactions on Power Systems*, vol. 34, no. 1, pp. 616–626, 2018.

VII. Appendix

A. Calculate partition function

For a multivariate Gaussian distribution $\boldsymbol{y} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\boldsymbol{\mu}$ denotes the mean and $\boldsymbol{\Sigma}$ denotes the covariance matrix, let $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$, we have:

$$\int_{\boldsymbol{y}} \sqrt{\frac{|\boldsymbol{\Lambda}|}{2\pi}} exp(-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^T \boldsymbol{\Lambda}(\boldsymbol{y}-\boldsymbol{\mu}))d\boldsymbol{y} = 1 \tag{22}$$

As mentioned earlier, the Gaussian CRF model $P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{x},\boldsymbol{\theta})} exp(\boldsymbol{\eta}^T\boldsymbol{y} - \frac{1}{2}\boldsymbol{y}^T\Lambda\boldsymbol{y})$ is equivalent to a multivariate Gaussian distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}, \boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$. Thus (22) can be rewritten as:

$$\sqrt{\frac{|\boldsymbol{\Lambda}|}{2\pi}} exp(-\frac{\mu^T\Lambda\mu}{2}) \int_{\boldsymbol{y}} exp(\boldsymbol{\eta}^T\boldsymbol{y} - \frac{1}{2}\boldsymbol{y}^T\Lambda\boldsymbol{y})d\boldsymbol{y} = 1 \tag{23}$$

Taking the nice properties of Gaussian distribution, the partition function $Z(\boldsymbol{x},\boldsymbol{\theta})$ can be calculated as:

$$Z(\boldsymbol{x},\boldsymbol{\theta}) = \int_{\boldsymbol{y}} exp(\boldsymbol{\eta}^T\boldsymbol{y} - \frac{1}{2}\boldsymbol{y}^T\Lambda\boldsymbol{y})d\boldsymbol{y} = \sqrt{\frac{2\pi}{|\Lambda|}} exp(\frac{\mu^T\Lambda\mu}{2}) \tag{24}$$

B. Surrogate loss

Mathematically, due to the Gaussian distribution properties, the original optimization problem in (17) is equivalently:

$$\min_{\boldsymbol{\theta}} \sum_{j=1}^{N} \frac{1}{2}(\boldsymbol{y^{(j)}} - \boldsymbol{\mu^{(j)}})^T \boldsymbol{\Lambda^{(j)}}(\boldsymbol{y^{(j)}} - \boldsymbol{\mu^{(j)}}) - \frac{1}{2} log|\boldsymbol{\Lambda^{(j)}}| \tag{25}$$

$s.t.$

$$\text{(forward pass) } \boldsymbol{\Lambda^{(j)}} = f_\Lambda(\boldsymbol{x^{(j)}}, \boldsymbol{\theta_\Lambda}), \forall j \tag{26}$$

$$\text{(forward pass) } \boldsymbol{\eta^{(j)}} = f_\eta(\boldsymbol{x^{(j)}}, \boldsymbol{\theta_\eta}), \forall j \tag{27}$$

$$\text{(positive definiteness) } \boldsymbol{\Lambda^{(j)}} \succ \boldsymbol{0}, \forall j \tag{28}$$

$$\text{(inference) } \boldsymbol{\mu^{(j)}} = \boldsymbol{\Lambda^{-1(j)}}\boldsymbol{\eta^{(j)}}, \forall j \tag{29}$$

To design a surrogate loss, we make an approximation $\boldsymbol{\Lambda^{(j)}} = \boldsymbol{I}$ ($\boldsymbol{I}$ is identity matrix) only in the objective function, so that $log|\boldsymbol{\Lambda}| = 0$ becomes negligible.