Physical Equation Discovery Using Physics-Consistent Neural Network (PCNN) Under Incomplete Observability

ABSTRACT

Deep neural networks (DNNs) have been extensively applied to various fields, including physical-system monitoring and control. However, the requirement of a high confidence level in physical systems made system operators hard to trust black-box type DNNs. For example, while DNN can perform well at both training data and testing data, but when the physical system changes its operation points at a completely different range, never appeared in the history records, DNN can fail. To open the black box as much as possible, we propose a Physics-Consistent Neural Network (PCNN) for physical systems with the following properties: (1) PCNN can be shrunk to physical equations for sub-areas with full observability, (2) PCNN reduces unobservable areas into some virtual nodes, leading to a reduced network. Thus, for such a network, PCNN can also represent its underlying physical equation via a specifically designed deep-shallow hierarchy, and (3) PCNN is theoretically proved that the shallow NN in the PCNN is convex with respect to physical variables, leading to a set of convex optimizations to seek for the physics-consistent initial guess for the PCNN. We also develop a physical rule-based approach for initial guesses, significantly shortening the searching time for large systems. Comprehensive experiments on diversified systems are implemented to illustrate the outstanding performance of our PCNN.

KEYWORDS

Physical System; Incomplete Observability; Physical Equation Discovery; Deep Neural Network; Convex Optimization

1 INTRODUCTION

With the era of the Internet of Things (IoT) coming to physical systems, there is an increasing need to extend monitoring and control to system edges, where traditional monitoring and control are unavailable. For example, power engineers nowadays try to provide a similar level of monitoring in its distribution grid when compared to the legendary transmission system with limited measurements. Under this situation, traditional system modeling are unavailable in many physical system edges, and machine learning (ML) tools are recognized as a viable way to conduct cost-efficient inferences for system operations in resource-limited areas, the topology estimation of power distribution systems [15, 32].

The mainstream of the ML methods for physical systems utilizes the universal approximation capacity in the deep neural networks (DNNs) to learn the data pattern [6]. However, for physical systems with an evolving operating point, this black box can't guarantee the model generalizability. To tackle this issue, sparse regression and symbolic regression [2, 3, 27] are introduced to recover the unknown physical equations, thus providing full model generalizability. In general, their methods utilize DNNs to create a base of physical symbols and introduce sparse regression to estimate the coefficients of these symbols. However, these methods usually assume the full observability of the system, i.e., the meters of the system are placed at every node. This assumption is often unrealistic due to the sensor cost. Considering the incomplete system observability, the above regression tools will fail since the hidden quantities with randomness create bias terms for a specific data set.

Thus, in this paper, we try to provide answers to the following question: can we maximize the recovery of physical system information while deploying the universal approximation capability in other unrecoverable areas? The problem is, in general, hard for arbitrary systems with even unknown bases for the system equation. However, for a large set of systems with clear physical quantities, the bases are known. For example, networks constrained via conservation laws [28] to deliver system flows like power (power systems), water (hydraulic networks), and kinetic energy (mass-damper systems). The conservation law further guarantees that the system equation parameters can be represented as a Laplacian matrix. These Laplacian systems have wide applications on resource delivery, finding consensus protocols of multi-agent networks [22], obtaining solutions of generalized flow problem [4], and characterizing systems' coupled oscillator motions [7].

To recover the physical equations of the incompletely-observable Laplacian systems and maximize the recovery of the physics, the key is to decompose a DNN model to the linear part with locally full observability and nonlinear part with high capacity to handle the randomness from hidden quantities. Thus, we propose a Physics-Consistent Neural Network (PCNN) with a deep-shallow structure to obtain maximal physical consistency. Firstly, PCNN can be shrunk to physical equations for sub-areas with full observability. Under this scenario, only the shallow NN in the PCNN activates. Secondly, PCNN creates some virtual nodes to represent an unobservable area. The deep NN in the PCNN activates in this situation to approximate the variables of the virtual nodes. Then, physical quantities related to the virtual nodes are constrained into a safe range and the local topology of the virtual nodes is guaranteed via controlling the sparsity of PCNN. In general, PCNN can therefore represent the physical equation of the reduced network. Finally, we theoretically prove that the shallow NN in the PCNN is *convex* with respect to physical variables, leading to a convex-optimization-based pre-training for the PCNN to provide the initial guess and save the training time. We have the following contributions.

- We introduce the problem of maximizing physical recovery for interpretation while minimizing approximating error in the non-recovery regions.
- We find the solution of PCNN to the problem above and provide theoretical guarantees for the PCNN for a physics-consistent solution.
- For speeding up the computation, the physical nodal categories are used to construct small-scaled but convexified problems to initialize PCNN.

- We conduct extensive experiments on diversified physical systems against many state-of-the-art models to demonstrate the superiority of our proposed methods.

2 RELATED WORK

2.1 Provide Interpretability of DNNs

Many kinds of research elaborate on how we can trust the DNNs via selecting important and explainable features (e.g., the input neurons) so that human users understand the selected features and decide if the trained DNN is trustworthy. These approaches can be categorized into the following groups. (1) The forward methods make perturbation of the specific input instance to evaluate the impact on the output. Such perturbation methods include occlusion [33], mutations [34], and input batch marginalization [35]. However, these methods are computationally expensive due to the large space of input perturbations. (2) The backward approaches are therefore proposed to only calculate the importance signal from an output neuron to the previous neurons. The importance signals typically include the gradient [25], layer-wise relevance [1], and difference-from-reference [24]. (3) Finally, an optimization-based method is also proposed to form an explanation model [18] with interpretable features to locally approximate the trained model. The feature weight represents its contributions for the user to evaluate. These methods in general lack the theoretical guarantee to interpret the physical system connectivity.

2.2 Improve the Generalizability of DNNs

To increase DNNs' generalizability and prevent overfitting, there are extensive methods like adding regularization terms, data augmentation, and early stopping. For model regularization, there are various penalty terms being introduced, e.g., Jacobian regularization [10], Kullback–Leibler divergence [31], weight matrix trace norm [13], and methods like dropout [26]. Another simple and useful approach is to conduct data augmentation [20] to increase the data size for training via flipping, rotation, scaling, cropping, translation, etc. Finally, implementing early stopping when training the DNN helps to avoid over-training and obtain a generalizable model [16]. Though these methods reduce the model complexity, they can hardly tackle the evolving physical systems with a continuously-changing operating point.

2.3 Enhance DNNs with Physics

Physical constraints in physical systems can improve the DNN performance. These works are typically categorized [30] into (1) modifying loss functions, (2) quantifying initialization point, (3) designing the DNN architecture, (4)modeling residual of traditional physical equations, and (5) implementing a hybrid physics-ML model. Specifically, adding a physics-based loss function essentially constrains some variables or parameters into a physical space, e.g., the law of energy conservation restricts the heat energy fluxes in the general lake model [11, 12]. Therefore, the parameter searching of a DNN will be more efficient, and the solution is reliable with the physical consistency. Introducing a physical initialization point can also easily reducing the training time. The obtaining the initialization usually following in a pre-training scheme with simulated data, e.g., the pre-training of a driving algorithm in a simulator [23].

Modeling the physical equation residuals or implementing hybrid models with both physical equations and ML models can be seen as the manipulation of physical equations and ML models like series connection [14] or parallel connection [29]. The physics-guided architecture design handles the issue via designing a structure that has interpretable neurons or connections into physical variables or connections. For example, [9] models the neuron connections as the line connections in power systems. Even though the above methods usually can't recover the complete physical equations. For physical equation recovery, the sparse regression and symbolic regression are usually introduced [2, 3, 27]. They typically utilize a DNN model to create bases for a physical equation. Further, a sparse linear regression is added to the DNN to select bases and estimate the coefficients of the bases.

3 PROBLEM FORMULATION

Many physical networks are graphs naturally, which can be modeled as a directed weighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with $\mathcal{V}$ to be the vertex set and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ to be the edge set. Based on the conservation law and system balance equations, the underlying physical equations of many physical systems can be formulated as [28]:

$$\dot{\boldsymbol{s}} = -\boldsymbol{L}\frac{\partial H}{\partial \boldsymbol{s}}(\boldsymbol{s}), \tag{1}$$

where $\boldsymbol{s}$ represents the storage at the vertices, and $\dot{\boldsymbol{s}}(t) = \frac{d\boldsymbol{s}(t)}{dt}$ represents the rate of the storage at the vertices, i.e., the net injected flows. $\boldsymbol{L}$ is the weighted Laplacian matrix of the graph $\mathcal{G}$ for the system physical parameters, and $H(\boldsymbol{s})$ represents the total stored energy at vertices. For example, in a hydraulic network, $\boldsymbol{s}$ can be the water volumes at each reservoir and $H(\boldsymbol{s})$ are the potential energies of the water.

Equation (1) is the foundation for all physical system analysis in this paper. For example, a subclass of equation (1), the power flow equations in the power systems, is the basis for power system planning, economic dispatch, stability analysis, and protection. Unfortunately, equation (1) is usually unknown for large systems, including both the system topology and the edge weights in the $\boldsymbol{L}$ matrix. This inspires the parameter estimation studies for the $\boldsymbol{L}$ matrix using sensor data in the system.

However, limited sensors in the system pose challenges for the estimation. Therefore, we denote $\mathcal{V} = \{\mathcal{O} \cup \mathcal{H}\}$, where $\mathcal{O}$ represents the observable node set and $\mathcal{H}$ represents the hidden node set. The measurements of $\boldsymbol{s}$, $\dot{\boldsymbol{s}}$, and $\frac{\partial H}{\partial \boldsymbol{s}}(\boldsymbol{s})$ can be metered or calculated for observable nodes. Formally, to represent these quantities, we denote $\boldsymbol{y} \in \mathcal{Y}$ and $\boldsymbol{x} \in \mathcal{X}$ as the random variables on the left hand side and the right hand side of (1), respectively. $\mathcal{Y} \subset \mathbb{R}^{n \times |\mathcal{V}|}$ and $\mathcal{X} \subset \mathbb{R}^{n \times |\mathcal{V}|}$ are the measurement spaces of $\boldsymbol{y}$ and $\boldsymbol{x}$, respectively, and $|\mathcal{V}|$ is the cardinality of $\mathcal{V}$. Then, we utilize the subscripts $\mathcal{H}$ and $\mathcal{O}$ as partitions of the variables/spaces and correspondingly denote $\boldsymbol{y}_{\mathcal{H}} \in \mathcal{Y}_{\mathcal{H}} \subset \mathbb{R}^{n \times |\mathcal{H}|}$, $\boldsymbol{y}_{\mathcal{O}} \in \mathcal{Y}_{\mathcal{O}} \subset \mathbb{R}^{n \times |\mathcal{O}|}$, $\boldsymbol{x}_{\mathcal{H}} \in \mathcal{X}_{\mathcal{H}} \subset \mathbb{R}^{n \times |\mathcal{H}|}$, and $\boldsymbol{x}_{\mathcal{O}} \in \mathcal{X}_{\mathcal{O}} \subset \mathbb{R}^{n \times |\mathcal{H}|}$. Based on the above definitions, we convert equation (1) into:

$$\begin{bmatrix} \boldsymbol{y}_{\mathcal{H}} \\ \boldsymbol{y}_{\mathcal{O}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{L}_{\mathcal{H}\mathcal{H}} & \boldsymbol{L}_{\mathcal{H}\mathcal{O}} \\ \boldsymbol{L}_{\mathcal{O}\mathcal{H}} & \boldsymbol{L}_{\mathcal{O}\mathcal{O}} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x}_{\mathcal{H}} \\ \boldsymbol{x}_{\mathcal{O}} \end{bmatrix}, \tag{2}$$

where we denote $\{\boldsymbol{x}_O^n\}_{n=1}^N \subset \mathcal{X}_O$ and $\{\boldsymbol{y}_O^n\}_{n=1}^N \subset \mathcal{Y}_O$ as the N samples obtained from meters. Thus, we focus on learning the mapping from $\boldsymbol{x}_O$ to $\boldsymbol{y}_O$ with the goal of finding a good mapping and recovering the parameters and the connectivity within nodes in O , i.e., $\boldsymbol{L}_{OO}$ for physical consistency. With the definition above, we have the following problem formulation for this paper.

- Problem: a data-driven physical consistent estimation that maximizes the physical recovery and approximation in the unrecoverable areas.
- Input: measurements $\{\boldsymbol{x}_O^n\}_{n=1}^N$ and $\{\boldsymbol{y}_O^n\}_{n=1}^N$ from observed nodes.
- Output: an accurate mapping f_θ such that $\boldsymbol{y}_O = f_\theta(\boldsymbol{x}_O)$. Further, partial of the parameters $\boldsymbol{\theta}_p \subset \boldsymbol{\theta}$ should accurately approximate the physical parameters, i.e., $\boldsymbol{\theta}_p \approx \boldsymbol{L}_{O,O}$ for physical consistence.

This is a non-trivial problem since (1) the hidden variables $\boldsymbol{x}_\mathcal{H}$ cause the systematic bias for the model, deteriorating the accurate topology and weight recovery within observed nodes, (2) there lacks theoretical guarantee of the approximation $\boldsymbol{\theta}_p \approx \boldsymbol{L}_{O,O}$, and (3) even when we have a good approximation, the model generalizability is hard to guarantee due to the randomness in $\boldsymbol{x}_\mathcal{H}$.

4 PROPOSED MODEL

The existence of the hidden nodes makes it difficult to directly identify the topology and the edge weights within observed nodes. Thus, we propose to separate the whole graph $\mathcal{G}$ into $|O|$ unit-graphs $\{\mathcal{G}_i = \{\mathcal{V}_i, \mathcal{E}_i\}\}_{i=1}^{|O|}$ with the graph center to be one observable node and radius to be 1, where we consider the distance between every two connected vertexes to be 1. We show that each unit graph can be approximated via an output channel of our PCNN model and provide the approximation guarantee in Section 5.

4.1 Graph Decomposition

In this subsection, we show different types of unit graphs for further constructing our PCNN model architecture.

Fully-observable unit-graph (F-Graph): This type of unit graph contains an observed node with all its 1-distance neighboring nodes observable. We denote the set of the central nodes in these unit graphs as $\mathcal{F}$. Therefore, any nodes $i \in \mathcal{F}$ with all of its 1-distance neighboring nodes, $Neigh(i)$, construct a fully-observable unit graph (F-Graph) $\mathcal{G}_i = \{i \cup Neigh(i), \mathcal{E}_i\}$. Then, the node i is *isloated* from $\mathcal{H}$. Based on equation (2), the topology and parameters of this sub graph can be accurately recovered via a linear layer of a neural network, i.e., a linear regression.

Partially-observable unit-graph (P-Graph): this type of area includes an observed node with at least one of its 1-distance neighboring nodes hidden. We denote the set of the central node set of these unit graphs as $\mathcal{P}$. Therefore, any node $j \in \mathcal{P}$ with nodes in $Neigh(j)$ construct a partially-observable sub graph (P-Graph) $\mathcal{G}_j = \{j \cup Neigh(j), \mathcal{E}_j\}$. Clearly, j has hidden boundary nodes $\mathcal{H} \cap Neigh(j)$. Thus, we need more layers instead of a linear layer to tackle the randomness from $\boldsymbol{x}_{\mathcal{H}\cap Neigh(j)}$, which requires multiple deep layers. Since the unknown $|\mathcal{H} \cap Neigh(j)|$ prevents the PCNN model construction, we aggregate $|\mathcal{H} \cap Neigh(j)|$ boundary nodes into K virtual nodes for $\mathcal{G}_i, \forall i \in \mathcal{P}$, where K is a hyper parameter. We show how to obtain a doable K value in Section 5. For $\mathcal{G}_j$, we denote the boundary virtual node set to be $\mathcal{N}_j = \{j_k\}_{k=1}^K$. **Notably, this modeling of P-Graph is an important reason why the proposed approach can be better than other methods.**

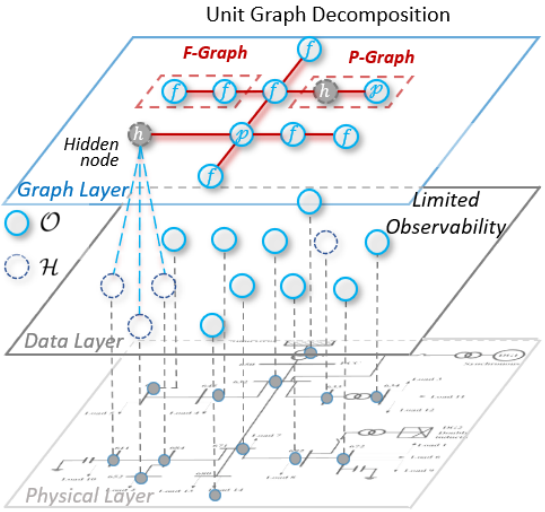


Figure 1: Physical system graph model and unit graph decomposition.

The graph decomposition inspires a deep-shallow design for the PCNN, as is shown in the following subsection.

4.2 PCNN Structure: Deep-Shallow Hierarchy

For each unit graph, the center node's output is the sum of all neighboring nodes' flow via the corresponding edge due to the law of conservation. Thus, the structure of the corresponding center node output channel is determined based on the unit graph type, together formulating the PCNN architecture. Specifically, we have the following designs.

F-Graph Layer: We utilize a linear layer to recover the topology and parameter of F-Graphs.

$$f_F(\boldsymbol{x}_O) = \boldsymbol{\theta}_F \boldsymbol{x}_O, \tag{3}$$

where $\boldsymbol{\theta}_F$ is the weights for the F-Graph layer. F-Graph layer needs to be pre-trained while other layers are frozen to identify the node set $\mathcal{F}$. The pre-training can also include a Lasso loss term $\lambda||\boldsymbol{\theta}_F||_0$ to guarantee sparsity, where λ is the hyper parameter for the penalty and $||\cdot||_p$ is the $l-p$ norm.

After the pre-training, we obtain $f_F^0(\boldsymbol{x}_O) = \boldsymbol{\theta}_F^0 \boldsymbol{x}_O$. $\forall i \in O$, the identification criteria is:

$$\frac{1}{N}\sum_{n=1}^{N} ||\boldsymbol{y}_O^n[i] - f_F^0(\boldsymbol{x}_O^n)[i]||_2 \leq \epsilon, \tag{4}$$

where $\boldsymbol{y}_O^n[i]$ and $f_F^0(\boldsymbol{x}_O^n)[i]$ are the i^{th} elements in $\boldsymbol{y}_O^n$ and $f_F(\boldsymbol{x}_O^n)$, respectively. ϵ is a hyper parameter. If equation (4) is satisfied, then $i \in \mathcal{F}$ since no hidden quantities contribute to the $\boldsymbol{y}_O[i]$. Based on the obtained $\mathcal{F}$ and $\mathcal{P} = O \setminus \mathcal{F}$, we have the following sequential **F-Graph initialization rule**: (1) $\forall i \in \mathcal{F}$, initialize $\boldsymbol{\theta}_F[i,:]$, the i^{th} row of $\boldsymbol{\theta}_F$, i.e., as the corresponding trained values $\boldsymbol{\theta}_F^0[i,:]$. (2) $\forall j \in \mathcal{P}, \forall i \in \mathcal{F}$, initialize $\boldsymbol{\theta}_F[j,i]$ as the trained value $\boldsymbol{\theta}_F^0[i,j]$. (3)

$\forall j \in \mathcal{P}, \forall i \in \mathcal{P}$, initialize $\boldsymbol{\theta}_F[j,i]$ as 0 if $i \neq j$ and $-\sum_{k \neq j} \boldsymbol{\theta}_F[j,k]$ if $i = j$.

For rule 2, $\forall j \in \mathcal{P}, \forall i \in \mathcal{F}$, if $ij \in \mathcal{E}$, the approximated $\boldsymbol{\theta}_F^0[i,j]$ for channel i is an accurate estimation of the true weight of line ij, while the estimation $\boldsymbol{\theta}_F^0[j,i]$ for channel j is inaccurate due to the hidden flows from hidden nodes to node j. If $ij \notin \mathcal{E}$, we have $\boldsymbol{\theta}_F^0[i,j] \approx 0$.

For rule 3, $\forall j \in \mathcal{P}, \forall i \in \mathcal{P}$, if $j \neq i$, the true weight of line ij can either be 0 when $ij \notin \mathcal{E}$ or not be accurately estimated in the F-Graph layer. Thus, we initialize $\boldsymbol{\theta}_F[j,i]$ to be 0 for both cases and estimate the weight of line ij in other layers of the PCNN. If $j = i$, the diagonal elements sum the weights with negative signs of all lines from j to $\mathcal{F} \cap Neigh(j)$ so that all the flows through these lines are well estimated. The only flows that need to be further explored lie in the edges to hidden nodes.

N-Approximation Layers: The initialization of F-Graph gives an accurate approximation of edge weights among nodes in the F-Graph. For edge weights in the P-Graph $\mathcal{G}_j$, **F-Graph initialization rule** 3 can't deliver an accurate parameter estimation due to the hidden quantities.

Thus, for $\mathcal{G}_j$, we model the contributions of hidden nodes via K virtual nodes $\mathcal{N}_j$ as mentioned before. Though the input samples of the virtual nodes are unknown, we can approximate them using the observed nodes' input and a deep neural network (N-Approximation Layers) f_N: $\boldsymbol{x}_{\mathcal{N}} = f_N(\boldsymbol{x}_O)$, where $\mathcal{N} = \bigcup_{j=1}^{|\mathcal{P}|} \mathcal{N}_j$ represents the total set of virtual nodes.

P-Graph Layer: For a P-Graph $\mathcal{G}_j$, the flows from nodes $\mathcal{F} \cap Neigh(j)$ are identified in the F-Graph Layer. Thus, we only need to consider the flows from $\mathcal{N}_j \cup Neigh(j)$. Since we know the measurements from $\mathcal{N}_j$, we build another linear layer (P-Graph Layer) such that:

$$\boldsymbol{y}_O - f_F(\boldsymbol{x}_O) = f_P(\boldsymbol{x}_{\mathcal{N}\cup O}) = \boldsymbol{\theta}_P \boldsymbol{x}_{\mathcal{N}\cup O}, \tag{5}$$

where $f_F(\boldsymbol{x}_O)$ represents the output from the F-Graph Layer, $\boldsymbol{x}_{\mathcal{N}\cup O} = [\boldsymbol{x}_{\mathcal{N}}; \boldsymbol{x}_O]$ is the concatenation of $\boldsymbol{x}_{\mathcal{N}}$ and $\boldsymbol{x}_O$, and $\boldsymbol{\theta}_P \in \mathbb{R}^{|O| \times |\mathcal{N}\cup O|}$ is the weight matrix of the P-Graph Layer.

Though for any node j, P-Graph Layer only models flows from $\mathcal{N}_j$, the variable at node j, $\boldsymbol{x}_O[j]$ is still utilized to calculate the flow from node j. Thus, we extend the mapping to the format of $f_P : \mathcal{X}^{|\mathcal{N}\cup O|} \to \mathcal{Y}^{|O|}$ to integrate the layer to the PCNN. We develop the following sequential **P-Graph initialization rule**: (1) $\forall j \in \mathcal{P}, \forall k \in \mathcal{N}$, if $k \in \mathcal{N}_j$ initializes $\boldsymbol{\theta}_P[j,k]$ from the optimal solution of a set of *convex optimizations* proposed in the next section. If $k \notin \mathcal{N}_j$, initialize $\boldsymbol{\theta}_P[j,k]$ to be 0. (2) $\forall j \in \mathcal{P}, \forall k \in \mathcal{N}, \forall i \in O$, if $j = i$ initialize $\boldsymbol{\theta}_P[j,i]$ to be $-\sum_{k \in \mathcal{N}} \boldsymbol{\theta}_P[j,k]$. If $j \neq i$, initialize $\boldsymbol{\theta}_P[j,i]$ to be 0. (3) $\forall i \in \mathcal{F}, \forall k \in \mathcal{N} \cup O$, initialize $\boldsymbol{\theta}_P[i,k]$ to be 0.

For rule 1, $\forall j \in \mathcal{P}, \forall k \in \mathcal{N}$, if $k \in \mathcal{N}_j$, the initial guess represents a good approximation for the weight of edge jk. In our next section, we propose a set of convex optimizations to obtain the optimal solution that both minimizes the squared loss and satisfies physical parameter constraints. We will theoretically prove that within these constraints, a globally optimal solution with zero loss for the noiseless data can be achieved due to convexity. If $k \notin \mathcal{N}_j$, edge jk does not exist so that the initial value of $\boldsymbol{\theta}_P[j,k]$ is 0. For rule 2, $\forall j \in \mathcal{P}, \forall k \in \mathcal{N}$ we sum the weight of jk (zero if jk does not exist) with a negative sign. Thus, the flow at line jk can be calculated. For rule 3, we don't consider the flows from $\mathcal{F}$ so that the related weights are set as 0s.

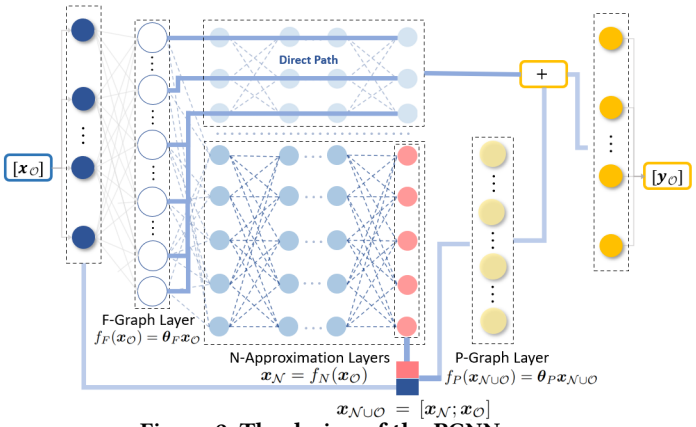


Figure 2: The design of the PCNN.

The optimization also brings good estimation values for $\boldsymbol{x}_{\mathcal{N}}$, thus inducing the **N-Approximation initialization rule**: Initialize parameters in $f_N(\boldsymbol{x}_O)$ via pre-training the network of f_N using input data from $\boldsymbol{x}_O$ and estimated data of $\boldsymbol{x}_{\mathcal{N}}$ from the proposed *convex optimizations* in the next section. In conclusion, we show our proposed PCNN model in Fig. 2. The formulation is as follows:

$$\boldsymbol{y}_O = f_{\boldsymbol{\theta}}(\boldsymbol{x}_O) = f_F(\boldsymbol{x}_O) + f_P([f_N(\boldsymbol{x}_O); \boldsymbol{x}_O]). \tag{6}$$

Though we have good initial parameters for the PCNN, the retraining of the PCNN as a whole is still required for an end-to-end optimization to minimize the total loss. Finally, the complete algorithm for the pre-training and the retraining process can be summarized in Algorithm 1.

Algorithm 1: Training Algorithm for PCNN

Function *Train-PCNN*

Input: Measurements $\{\boldsymbol{x}^n\}_{n=1}^N$ and $\{\boldsymbol{y}^n\}_{n=1}^N$ from observed nodes, threshold ϵ;
Pre-train the F-Graph Layer using Lasso regression;
Obtain $\mathcal{F}$ set using criteria (4) with ϵ;
$\mathcal{P} = O \setminus \mathcal{F}$;
Initialize $\boldsymbol{\theta}_F$ using **F-Graph initialization rule**;
Solve the proposed optimization in equation (7) using Algorithm 2;
Initialize $\boldsymbol{\theta}_P$ using **P-Graph initialization rule**;
Initialize parameters in the deep layers $f_N(\boldsymbol{x}_O)$ using **N-Approximation initialization rule**;
Retrain PCNN using BP algorithm;
Output: PCNN model;

end

5 PHYSICS-CONSISTENT INITIALIZATION USING CONVEX OPTIMIZATION

The proposed PCNN embraces the deep-shallow structure where the deep NNs approximate the hidden variable, and the shallow NN formalizes all the variables into the physical-equation representation. Specifically, each output channel represents a nodal balance

equation. In this section, we verify that our initialization rules for the F-Graph and P-Graph Layers can provide a physics-consistent solution. We first define this solution as follows.

Definition 5.1 (Physics-consistent solution)**.** A physics-consistent solution (PCS) for the weights of F-Graph and P-Graph Layers and the output of the N-Approximation Layer brings 0 loss for the noiseless data and lie within the physically constrained region.

Based on the definition, the solutions for the F-Graph Layer can be obtained via a linear Ordinary Least Square (OLS) to optimize $\boldsymbol{\theta}_F$ in (3). The solutions for the P-Graph Layer and the output of the N-Approximation Layer can be obtained via the following optimization.

Specifically, we treat one P-Graph $\mathcal{G}_j, \forall j \in \mathcal{P}$ as an example. Based on our initialization rules, the weight of $ji, \forall i \in \mathcal{F} \cap Neigh(j)$ has been quantified in the F-Graph Layer. Thus, we only need to discuss the weight of $jk, \forall k \in \mathcal{N}_j \cap Neigh(j)$. We denote w_k as the weight of line jk, and w_k is one element in the parameter matrix $\boldsymbol{\theta}_P$ in the P-Graph Layer. Further, we let $p_n := \boldsymbol{y}_O^n[j] - f_F^0(\boldsymbol{x}_O^n)[j]$ as the n^{th} sample net flow from $\mathcal{N}_j$, where the estimation of the net flow is guaranteed via accurate parameter estimation of the F-Graph Layer. Similarly, we let $x^n := \boldsymbol{x}_O^n[j]$ as the n^{th} input measurement of node j. Finally, we denote x_k^n as the n^{th} approximated nodal measurements for the k^{th} virtual node in $\mathcal{N}_j$. Thus, x_k^n is a realization of one element in $\boldsymbol{x}_{\mathcal{N}}^n$. Here we eliminate the index j in w_k, p_n, x^n and x_n^k for simplicity. To find a good physics-consistent initialization, we propose to treat w_k and x_n^k as *variables* and formalize the following optimization $\mathbb{P}_K^j$.

$$\begin{aligned} \min_{w_k, x_k^n} L &= \sum_{n=1}^{N} (p_n - \sum_{k=1}^{K} w_k (x^n - x_k^n))^2 \\ s.t.\ \ &\{w_k, \{x_k^n\}_{n=1}^N\}_{k=1}^K \in C_K, \end{aligned} \tag{7}$$

where L is the loss and we eliminate the index j, K for simplicity. C_K represents for K virtual nodes, the feasible region under a set of physical constraints. For example, the tolerance of the nodal devices requires x_k^n to have positive minimum and maximum values. Further, the capacity of the line jk limits the maximum values of the flow on that line, i.e., $|w_k(x^n - x_k^n)|$ has an upper bound. It can be easily proven that under the above constraints, C is convex. We assume this convexity holds in general for all the proposed physical constraints.

To prove the PCS can be obtained via the above optimizations, we prove the existence of the PCSs in the subsequent section.

5.1 Existence of the solution

Proof. F-Graph Layer has the feasible solution of the ground-truth physical grid parameters.

As for P-Graph Layer and N-Approximation Layer, we denote the PCS as $\{\bar{w}_k, \{\bar{x}_k^n\}_{n=1}^N\}_{k=1}^K$ in equation (7). For node j, we assume there are M number of true hidden nodes connecting j with line jm parameter as $b_m, 1 \le m \le M$ and true input nodal measurements as $x_m^n, \forall 1 \le n \le N$ for the n^{th} sampling time. Since we assume the PCS produces 0 loss, we have the following equations:

$$\begin{aligned} &\sum_{k=1}^{K} \bar{w}_k (x^n - \bar{x}_k^n) = \sum_{m=1}^{M} b_m (x^n - x_m^n), \forall 1 \le n \le N \\ &s.t.\ \ \{\bar{w}_k, \{\bar{x}_k^n\}_{n=1}^N\}_{k=1}^K \in C_K. \end{aligned} \tag{8}$$

It's obvious that when $M = K$, $\{b_m, \{x_m^n\}_{n=1}^N\}_{m=1}^M$ is a PCS. However, since equation (8) is under-determined, multiple PCSs exist within C_K, even when $K \neq M$. To find one of these solutions, we show in the next subsection that the problem $\mathbb{P}_K^j$ is convex under certain assumptions, and we can iteratively increase K and solve $\mathbb{P}_K^j$ to obtain one PCS. □

5.2 Model Convexity

We have the following proofs for the model convexity for the pre-training of the F-Graph and P-Graph Layers.

Proof. For F-Graph Layer, the pre-training is an ordinary least square optimization, which implies convexity.

For P-Graph Layer, the optimization $\mathbb{P}_K^j$ is in (7). Since C_K is convex, we only need to consider the convexity of the loss function. Thus, We construct the Hessian matrix of the loss function with the following elements:

$$\frac{\partial L}{\partial^2 w_k} = \sum_{n=1}^{N} 2(x_k^n - x^n)^2,\ \ \frac{\partial L}{\partial w_k \partial w_h} = \sum_{n=1}^{N} 2(x_h^n - x^n)(x_k^n - x^n),$$

$$\frac{\partial L}{\partial w_k \partial x_k^n} = 2(p_n - \sum_{l=1}^{K} w_l (x^n - x_l^n) - w_k (x^n - x_k^n)),\ \ \frac{\partial L}{\partial^2 x_k^n} = 2w_k^2,$$

$$\frac{\partial L}{\partial w_k \partial x_h^n} = 2(x_k^n - x^n) w_h,\ \ \frac{\partial L}{\partial x_k^n x_h^n} = 2w_k w_h,\ \ \frac{\partial L}{\partial x_k^n x_h^m} = 0.$$

We study the positive-definiteness of the Hessian matrix $\boldsymbol{H}_0$ with respect to the variable vector $[w_1, \cdots, w_K, x_1^1, x_2^1 \cdots, x_K^1, x_1^2, \cdots, x_K^N]^T$. It's clear that $\boldsymbol{H}_0[1:K, 1:K]$ is positive semi-definite, since this Hessian matrix $\boldsymbol{H}_0[1:K, 1:K]$ represents a linear least square loss. On the other hand, if we conduct a Gaussian elimination process to iteratively prove the positive semi-definiteness, we need to iteratively prove the first entry of each eliminated matrix is positive. Due to the positive semi-definiteness of $\boldsymbol{H}_0[1:K, 1:K]$, it's obvious that during the first $K-1$ eliminations, all the first entries of the eliminated matrices are positive, i.e., $\boldsymbol{H}_1(1,1), \cdots, \boldsymbol{H}_{K-1}(1,1) > 0$. Thus, we focus on the impacts of eliminations on diagonal entries after the previous K numbers.

Specifically, for the i^{th} Gaussian elimination, we can evaluate the diagonal element of the eliminated matrix as:

$$\boldsymbol{H}_i(l,h) = \boldsymbol{H}_{i-1}(l+1, h+1) - \frac{\big(\boldsymbol{H}_{i-1}(1, h+1)\big)\big(\boldsymbol{H}_{i-1}(1, l+1)\big)}{\boldsymbol{H}_{i-1}(1,1)}, \tag{9}$$

where $1 \le l \le (N+1)K - i$.

If we assume N is sufficiently large, $\mathbf{H}_0(k,k) = \sum_{n=1}^{N} 2(x_k^n - x^n)^2, \forall 1 \le k \le K$. The non-zero flow of line jk (recall j is the index of the center node of $\mathcal{G}_j$) implies $x_k^n - x^n \neq 0$. Namely, $\mathbf{H}_0(k,k)$ is a sufficiently large positive number. More specifically, equation (5.2) implies that $\forall b \ge K, \boldsymbol{H}_0(1, 1+b) \ll \boldsymbol{H}_0(1,1)$. Therefore, the elimination process in equation (9) indicates that $\boldsymbol{H}_1(b,b) > 0$. However, we need to consider the values of $\boldsymbol{H}_1(1,b)$ and $\boldsymbol{H}_1(1,1)$ to

continue the iteration. Due to the triangle inequality, we know that $\boldsymbol{H}_1(1,1) > 0$ given $x_1^n \neq x_2^n$ for any $1 \leq n \leq N$. If N is sufficiently large, we can claim that $\boldsymbol{H}_1(1,1)$ has a sufficiently large positive accumulation value, compared to a fixed value of $\boldsymbol{H}_1(1,b)$. Thus, we have $\boldsymbol{H}_1(1,1) \gg \boldsymbol{H}_1(1,b)$. Repeating the above eliminations for K times and we have $\boldsymbol{H}_K(b+1-K, b+1-K) > 0$.

Then, for the rest of $b-K$ eliminations, the diagonal element $\boldsymbol{H}_{K+a}(1,1), \forall 1 \leq a \leq b-K$ is not sufficiently large. However, since the off-diagonal $\boldsymbol{H}_{K+a}(1, b+1-(K+a)) = 0$ always hold during the Gaussian eliminations, we can still guarantee $\boldsymbol{H}_{K+a}(b+1-(K+a), b+1-(K+a)) > 0$. Finally, the elimination will end up with $\boldsymbol{H}_b(1,1) > 0$.

In general, the above iteration process proves the positivity of each first entry of the eliminated matrices, indicating the positive semi-definiteness of the Hessian matrix and the convexity of our problem $\mathbb{P}_K^j$. □

5.3 Model solving algorithm

The process above presents for a P-Graph $G_j, \forall j \in \mathcal{P}$, the existence of the PCS for some Ks and the convexity of $\mathbb{P}_K^j$ for any K. Thus, we propose to iteratively solve $\mathbb{P}_K^j$ and evaluate if the solution is a PCS. Since the real-world data is not noiseless, we employ a threshold ϵ_1 for the evaluation. Then, the algorithm is shown in Algorithm 2 for the PCS for P-Graph Layer and N-Approximation Layer.

Algorithm 2: Training Algorithm for $\{\mathbb{P}_K^j\}, \forall j \in \mathcal{P}$

Function *Train-*$\mathbb{P}_K^j$
  Input: Measurements $\{\boldsymbol{x}^n\}_{n=1}^N$ and $\{\boldsymbol{y}^n\}_{n=1}^N$ from observed nodes, threshold ϵ_1, and $\mathcal{P}$;
  $K = 1$;
  forall *j=1 to* $|\mathcal{P}|$ **do**
    while $L_K^j > \epsilon_1$ **do**
      Use gradient descent method to solve $\mathbb{P}_K^j$;
      Evaluate L_K^j, i.e., the loss of $\mathbb{P}_K^j$;
      $K = K + 1$;
    end
  end
  Obtain the PCS as the optimal solutions of the above optimizations;
  Output: A PCS for weights in the P-Graph Layer and outputs of the N-Approximation Layer;
end

6 EXPERIMENT

6.1 Dataset Description

In our experiment, we introduce power systems, mass-damper systems, hydraulic networks, and the graph of large systems from the University of Florida (UF) sparse matrix collection [5] as the underlying physical system for model training and comparison. Specifically, the dataset descriptions are as follows.

IEEE Power Systems and PJM Load Data. IEEE provides standard power system models, including the grid topology, parameters, and generation models, etc., for accurate simulations on the power domain. The model files and the simulation platform, MATPOWER [19], are based on MATLAB. In this experiment, we incorporate IEEE 19-, 30-, 57-, 69-, and 85-systems for testing. To conduct the simulation, the load files are required as the input to the systems. Thus, we introduce real-world power consumptions in PJM Interconnection LLC (PJM) data [21]. The load files contain hourly power consumption in 2017 for the PJM RTO regions. With the above data, MATPOWER produces the system states of voltage angle $\boldsymbol{\phi}$ and system input active power flow $\boldsymbol{p}$, indicating the linearized power flow equations $\boldsymbol{p} = \boldsymbol{L}_A\boldsymbol{\phi}$, a special case for the general physical system formulation in (1), where $\boldsymbol{L}_A$ is the weighted Laplacian matrix (i.e., the susceptance matrix) of the electric system.

Mass-damper system data. The mass-damper systems can be represented with the physical equation $\dot{\boldsymbol{q}} = -\boldsymbol{D}\boldsymbol{R}\boldsymbol{D}^\top\boldsymbol{M}^{-1}\boldsymbol{q}$, where $\boldsymbol{q}$ is the vector of momenta of the masses, $\boldsymbol{D}$ is the incidence matrix of the graph, $\boldsymbol{R}$ is the diagonal matrix of the damping coefficients of the damper attached to the edges, and $\boldsymbol{M}$ is the diagonal mass matrix [28]. Using MATLAB, we simulate the dynamic process of the mass-damper system with 10 buses and obtain $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$.

UF sparse matrix-based system. The UF sparse matrix collection provides a lot of large sparse matrix-based networks. In this experiment, we utilize the 2003-bus system to test.

Therefore, we have three different systems, providing testing on 10-, 19-, 57-, 69-, 85-, and 2003-node networks. To consider different system observability, we change the ratio of the number of the observed nodes to that of the total nodes $\gamma \in \{0.1, \cdots, 0.9\}$.

6.2 Benchmark Models

To fully investigate the strong interpretability and generalizability of our PCNN, we compare our proposed PCNN with other advanced DNN models. Specifically, we have the following benchmark models for comparison.

- **Resnet** [8]. Deep Residual Network creates a shortcut connection to pass the deep information directly to the shallow layers. Such a skip-connection effect not only helps to avoid gradient vanishing issues in the training phase, but also contributes to the model generalization ability since the low-complexity features are connected to the output, thus decreasing the model complexity [17].
- **SINDYs** [2, 3]. The sparse identification of nonlinear dynamics (SINDy) utilizes the sparse regression technique to recover the parameters of the physical systems, while the base of the regression can be selected via DNNs. In our experiments, we consider systems with a fixed symbolic base (i.e., we know $H(\boldsymbol{s})$ in equation (1)) is fixed due to the prior knowledge, and we eliminate the DNN part for simplicity.
- **DNNs with Dropout Method** [26]. Dropout method randomly disables neurons in training, thus preventing the neurons from over co-adapting and increasing the model generalizability.
- **DeepLIFT** [18, 24]. DeepLIFT is an advanced model to select important features of a well-trained DNN via calculating the importance signals from output to the input features. In this experiment, we calculate the SHAP (SHapley Additive exPlanations) values of features in the trained DNN via DeepLIFT

[18]. Thus, we can select the important features and evaluate the model interpretability.

6.3 Model Evaluations

We propose the following metrics to evaluate the generalizability and interpretability of PCNN and benchmark models.

Generalizability. We conduct 5-fold cross-validation to evaluate the model generalizability. Mean square error (MSE) of the validation set is used to evaluate the model performance of predicting $\boldsymbol{y}_O$.

Parameter estimation. We utilize all the data to estimate the system parameters. To evaluate the model performance, we consider the following two aspects: (1) for lines among node set O, both the line weight estimation error and the connectivity should be evaluated. Since the connectivity can be converted to the sparsity of the Laplacian matrix, we utilize the so-called normalized Total Vector Error ($nTVE$) [15] to evaluate the difference between the estimated $\hat{\boldsymbol{L}}$ and the true Laplacian matrix $\boldsymbol{L}$:

$$nTVE = 100 \times \frac{||\hat{\boldsymbol{L}} - \boldsymbol{L}||_2}{||\boldsymbol{L}||_2}. \tag{10}$$

(2) for lines between O to $\mathcal{N}$, the connecivity is known so we use percent difference (PE) to evaluate the error between the estimated $\hat{w}$ and the true weight w for one line: $PE = 100 \times \frac{\hat{w}-w}{w}$.

Interpretability. The model interpretability determines the critical input features with respect to each output channel. For Resnet and DNNs with Dropout method, we utilize DeepLIFT [18] for important feature selection. For our proposed PCNN, the sparsity of $\boldsymbol{\theta}_F$ illustrates the estimated topology within O. Thus, for each $i \in O$, the inputs in the neighboring nodes in the *estimated unit graph* are the important features. For the SINDy method, we denote input features with non-zero coefficients for one output feature as its important features.

In general, we denote the indices of the estimated important features as $Import(i)$ for the i^{th} output and the ground true indices are $Neigh(i) \cup \{i\}$. Thus, we introduce the measure $h(\%)$:

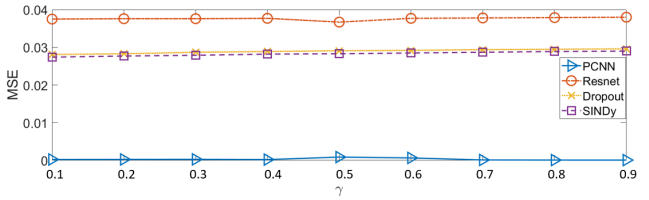
$$h = \sum_{j \in O} \frac{J(Import(j), Neigh(j) \cup \{j\})}{|O|} \times 100\%, \; J(X, Y) = \frac{|X \cap Y|}{X \cup Y},$$

where $J(\cdot, \cdot)$ is the so-called Jaccard index.

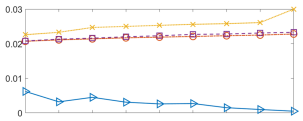
6.4 Results for Model Generalizability

In our experiments, we test different systems with changing γ to comprehensively compare the model generalizability among different methods. 5-fold cross-validation is conducted. The results are shown in Fig. 3a to 3g. We find that for each trial, our PCNN **always achieves the lowest MSE value** in the validation dataset. Further, the MSE of our PCNN decreases as γ increases, while for other methods, the MSE increases.

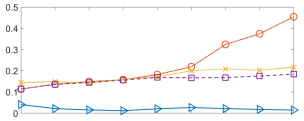
The lowest generalization error comes from (1) the well-extracted local governing equations that are generalizable to different datasets and (2) the physical constraints that enable the physical variables to be within the physical range. Secondly, the increasing of sensor penetration (γ) leads to more physical parameters to be captured,



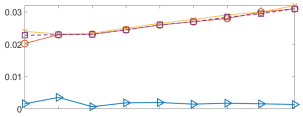
(a) MSE for different methods in the 10-bus system.



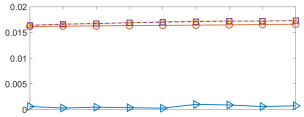
(b) MSE for different methods in the 19-bus system.



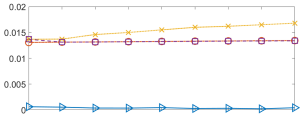
(c) MSE for different methods in the 30-bus system.



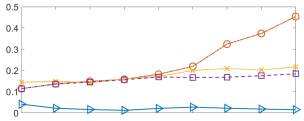
(d) MSE for different methods in the 57-bus system.



(e) MSE for different methods in the 69-bus system.



(f) MSE for different methods in the 85-bus system.



(g) MSE for different methods in the 2003-bus system.

Figure 3: The MSE value for different testing systems.

thus decreasing the MSE further. However, for other methods without physical consistency, MSE will increase due to the growth of the output dimensionality.

6.5 Results for Network Parameter Estimation

In this subsection, we show the second function of our PCNN, i.e., estimating the edge weights to construct the underlying physical equations. For the line parameters and connectivity among observed nodes O, we calculate the $nTVE(\%)$ for evaluation. The comparison is between our PCNN and the SINDy since other DNNs can't estimate the physical equation parameters.

The result is shown in Table 1. Generally, our PCNN far outperforms the SINDy method for all systems when $\gamma < 0.5$. Empirically, the PCNN's $nTVE$ is around $10\% \sim 25\%$ of the SINDy's $nTVE$. When γ increases, the performance of the PCNN and SINDy will become closer. However, PCNN's $nTVE$ still only covers around $40\% \sim 60\%$ of SINDy's $nTVE$. The reasons are as follows. (1) PCNN employs a testing criterion in equation (4) to decompose O into $\mathcal{F}$ and $\mathcal{P}$. Then, the initialization rule of the PCNN can enable the shared weights between $\mathcal{F}$ and $\mathcal{P}$ to always be accurately estimated in the pre-training of the F-Graph. For the SINDy method, however, the shared weight estimation incurs errors due to hidden quantities. (2) when $\gamma < 0.5$, the hidden nodes are dominant so that PCNN performs much better than SINDy. (3) when γ is increasing, the number of hidden nodes decreases so that the inaccurate estimation of the shared weights in SINDy decreases, forcing PCNN and SINDy

Table 1: $nTVE(\%)$ error of parameter estimation for PCNN and SINDy methods.

	10-bus		19-bus		30-bus		57-bus		69-bus		85-bus		2003-bus	
γ	PCNN	SINDy	PCNN	SINDy	PCNN	SINDy	PCNN	SINDy	PCNN	SINDy	PCNN	SINDy	PCNN	SINDy
0.1	69	381	4.5	23	32	89	73	169	65	648	31	139	89	399
0.2	51	317	3.6	21	33	83	65	198	68	723	34	121	74	421
0.3	56	265	6.3	24	30	81	78	156	68	614	24	118	61	406
0.4	43	198	3.3	18	27	78	71	153	54	598	19	123	59	385
0.5	45	118	2.9	15	27	72	72	145	79	470	16	121	78	335
0.6	62	97	4.4	12	23	61	64	132	76	423	13	104	66	299
0.7	41	65	0.95	7.3	12	55	52	122	69	327	9.8	96	64	301
0.8	37	72	0.89	5.1	8.8	29	47	98	43	211	10	93	59	276
0.9	18	32	0.73	4.8	9.5	21	34	94	22	108	10	71	57	283

to have closer performance. Secondly, we observe in Table 1 that for 19-bus system, PCNN and SINDy have relatively small $nTVE$ compared to other systems. This is because 19-bus system is radial so that a hidden node will only cause errors within one line for line parameter estimation.

Finally, we study the weight estimation between nodes in $\mathcal{P}$ and $\mathcal{N}$. Essentially, our PCNN gives an equivalent estimation within the physical ranges to create an equivalent network to the true network. This equivalence is *not the same* as the ground truth. Thus, the error calculation for lines between $\mathcal{P}$ and $\mathcal{N}$ is generally meaningless. However, intuitively, the equivalent network will be closer and closer to the true network when the prior physical constraints are smaller and smaller in the PCNN model. The above trend is worth studying to numerically illustrate the PCNN's improvement when knowing more knowledge of the physical constraints.

Specifically, we utilize 19-bus system as an example. As is shown in Fig. 4, we target at the estimation of w_1 and w_2. The true values are 26.42 and 10.89, respectively. Then, we gradually decrease the physical ranges of w_1 and w_2 from $[0, 100]$ and $[0, 100]$ to $[25, 30]$ and $[10, 15]$ and calculate the $PE(\%)$ errors.

The result is shown in Fig. 4. In the x-axis, we set the base area to be $S_b = 100$ and use the ratio of $\frac{S}{S_b}$ to represent the level of physical constraints for a constraint area S. In the y-axis, we plot the $PE(\%)$ error for w_1 (green square) and w_2 (blue circle). We find that when the ratio is decreasing, the error will decrease. More specifically, When the ratio is larger than 60%, the errors in most of the testing scenarios are higher than 40%. When the ratio is less than 10%, the errors are less than 21%. This region can be a good indicator of how much we need to know about the prior to enabling the estimation to approach the ground truth. Finally, we find that there are points when the ratio is high while the error is small. They are caused by chance when we do the optimization in equation (7) and randomly choose the initial points of w_1 and w_2 that are close to the true points.

6.6 Results for Model Interpretability

To test the model interpretability, we set $\gamma = 0.5$ and calculate the measure h in (6.3) under different scenarios, as is shown in Table 2. Our PCNN can always obtain 100% interpretable features, which show that the estimated topology within O is correct. The perfect

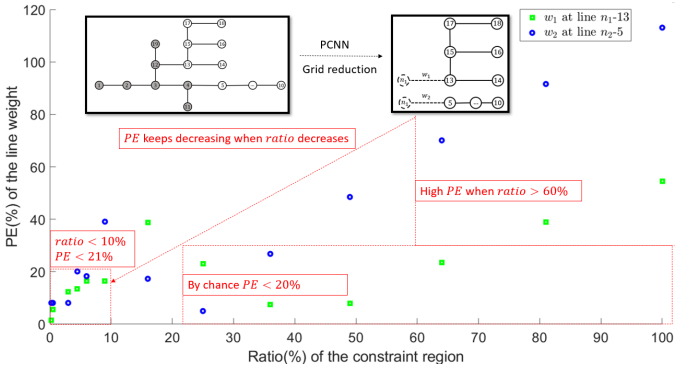


Figure 4: The $PE\%$ of the line weight estimation with respect to different physical constraint areas.

Table 2: The $h(\%)$ value for different methods in different systems.

	PCNN	SINDy	Resnet	Dropout
10-bus	100	100	74.0	52.4
19-bus	100	93.2	69.0	36.7
30-bus	100	92.9	71.9	31.3
57-bus	100	85.3	57.4	12.8
85-bus	100	85.7	73.8	23.8
2003-bus	100	75.4	73.8	23.8

performance essentially comes from the sparsity control when pre-training the F-Graph Layer. For SINDy method, the sparsity control also exists, thus yielding high h values. However, the hidden quantities bring some incorrect connectivity and prevent the h to be 100%. For the other two DNNs, h will decrease about 30% $\sim$ 80% due to the complex correlations in the NN model.

6.7 Results of Dynamic Simulation

We further demonstrate the quality of the estimated virtual grid from PCNN via implementing the dynamic simulation for the ground-truth grid and the virtual grid. Specifically, we utilize the 10-bus mass-damper system as an example. The ground-truth grid and the virtual grid are shown in Fig. 5. We set the initial velocity $\boldsymbol{x}^0 \in \mathbb{R}^{10\times 1}$ for the ground-truth grid. Then, we input the data of $\boldsymbol{x}^0_O$

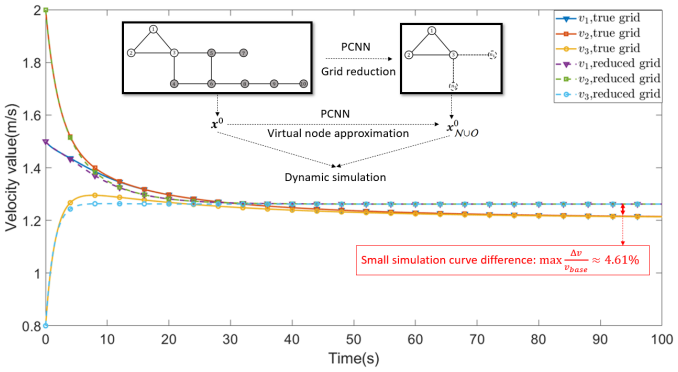


Figure 5: The dynamic simulation using the ground-truth grid and the virtual grid from PCNN.

to the well-trained PCNN and output $x^0_{\mathcal{N}}$ in the N-Approximation Layer, where $O = \{1, 2, 3\}$ and $\mathcal{N} = \{n_1, n_2\}$ in this experiment. Subsequently, we can conduct the dynamic simulation for the two grids, where the virtual grid's parameters are learned from the PCNN. The result is shown in Fig. 5. We find that the simulation curve has an overall small difference and the ratio of the maximum velocity difference to the base velocity ($1m/s$) is 4.61%.

7 CONCLUSION

We propose a Physics-Consistent deep Neural Network (PCNN) to discover physical equations for the Laplacian systems under incomplete observability. PCNN can be shrunk to physical equations automatically for fully-observable areas, reduce hidden nodes to virtual nodes to create a reduced grid, and maintain the physical ranges and topology of the reduced grid with the deep-shallow PCNN structure and physical constraints. Finally, we provide a theoretical guarantee to find a good initial guess of the PCNN to save the searching time. Extensive experiments are conducted in the power, mass-damper, and UF sparse matrix systems.

REFERENCES

[1] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one* 10, 7 (2015), e0130140.

[2] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. 2016. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences* 113, 15 (2016), 3932–3937.

[3] Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. 2019. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences* 116, 45 (2019), 22445–22451.

[4] Samuel I Daitch and Daniel A Spielman. 2008. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*. 451–460.

[5] Timothy A Davis and Yifan Hu. 2011. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)* 38, 1 (2011), 1–25.

[6] Jonathan Goh, Sridhar Adepu, Marcus Tan, and Zi Shan Lee. 2017. Anomaly detection in cyber physical systems using recurrent neural networks. In *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*. IEEE, 140–145.

[7] Aric Hagberg and Daniel A Schult. 2008. Rewiring networks for synchronization. *Chaos: An interdisciplinary journal of nonlinear science* 18, 3 (2008), 037105.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[9] Xinyue Hu, Haoji Hu, Saurabh Verma, and Zhi-Li Zhang. 2020. Physics-Guided Deep Neural Networks for PowerFlow Analysis. *arXiv preprint arXiv:2002.00097* (2020).

[10] Daniel Jakubovitz and Raja Giryes. 2018. Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 514–529.

[11] Xiaowei Jia, Jared Willard, Anuj Karpatne, Jordan Read, Jacob Zwart, Michael Steinbach, and Vipin Kumar. 2019. Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles. In *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 558–566.

[12] Xiaowei Jia, Jared Willard, Anuj Karpatne, Jordan S Read, Jacob A Zwart, Michael Steinbach, and Vipin Kumar. 2020. Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles. *arXiv preprint arXiv:2001.11086* (2020).

[13] Y. Jiang, Z. Wu, J. Wang, X. Xue, and S. Chang. 2018. Exploiting Feature and Class Relationships in Video Categorization with Regularized Deep Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 2 (2018), 352–364.

[14] Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. 2017. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431* (2017).

[15] Haoran Li, Yang Weng, Yizheng Liao, Brian Keel, and Kenneth E Brown. 2021. Distribution grid impedance & topology estimation with limited or no micro-PMUs. *International Journal of Electrical Power & Energy Systems* 129 (2021), 106794.

[16] Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. 2020. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 4313–4324.

[17] Yin Liu and Vincent Chen. 2018. On the Generalization Effects of DenseNet Model Structures. (2018).

[18] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*. 4765–4774.

[19] MATPOWER community. 2020. MATPOWER. (2020). https://matpower.org/.

[20] Agnieszka Mikołajczyk and Michał Grochowski. 2018. Data augmentation for improving deep learning in image classification problem. In *2018 international interdisciplinary PhD workshop (IIPhDW)*. IEEE, 117–122.

[21] PJM Interconnection LLC. 2018. Metered Load Data. (2018). https://dataminer2.pjm.com/feed/hrl_load_metered/definition.

[22] R. O. Saber and R. M. Murray. 2003. Consensus protocols for networks of dynamic agents. In *Proceedings of the 2003 American Control Conference, 2003.*, Vol. 2. 951–956. https://doi.org/10.1109/ACC.2003.1239709

[23] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. 2018. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*. Springer, 621–635.

[24] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685* (2017).

[25] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).

[26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[27] Silviu-Marian Udrescu and Max Tegmark. 2020. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances* 6, 16 (2020), eaay2631.

[28] Arjan van der Schaft. 2017. Modeling of physical network systems. *Systems & Control Letters* 101 (2017), 21–27.

[29] Zhong Yi Wan, Pantelis Vlachas, Petros Koumoutsakos, and Themistoklis Sapsis. 2018. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PloS one* 13, 5 (2018), e0197704.

[30] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. 2020. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919* (2020).

[31] Dong Yu, Kaisheng Yao, Hang Su, Gang Li, and Frank Seide. 2013. KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 7893–7897.

[32] J. Yu, Y. Weng, and R. Rajagopal. 2017. Robust mapping rule estimation for power flow analysis in distribution grids. In *2017 North American Power Symposium (NAPS)*. 1–6.

[33] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*. Springer, 818–833.

[34] Jian Zhou and Olga G Troyanskaya. 2015. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods* 12, 10 (2015), 931–934.

[35] Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. 2017. Visualizing deep neural network decisions: Prediction difference analysis. *arXiv preprint arXiv:1702.04595* (2017).