# OptNet-Embedded Data-Driven Approach for Optimal Power Flow Proxy

Yixiong Jia, Yiqin Su, Chenxi Wang, Yi Wang

Abstract—Solving AC-optimal power flow (AC-OPF) in real-time is crucial for further power system operation and security analysis. To this end, data-driven methods are employed to directly output the OPF solution. However, due to the prediction error, it is a challenge for data-driven methods to provide a feasible solution. To address this issue, different feasibility-enhanced methods are proposed. However, they are either computationally expensive or cannot provide feasible solutions. In this paper, we propose an OptNet-embedded data-driven approach for AC-OPF proxy to provide a feasible solution efficiently. This approach designs a three-stage neural network architecture to represent the OPF problem, where the first stage is used to lift the dimension of the input, the second stage is used to approximate the OPF problem using OptNet, and the third stage is used to decouple the high-dimensional solution to acquire the OPF solution. Finally, to expedite the solving process, a two-step pruning method is proposed to remove the unnecessary inequality constraints and values. Numerical experiments on the IEEE 4- and 14-bus test systems validate that the proposed approach can provide a "good enough" feasible solution.

Index Terms—Optimal power flow, OptNet layer, model pruning, data-driven, power system

## I. Introduction

THE optimal power flow (OPF) is a fundamental tool for power system operation and security analysis [1]. Traditionally, the interior-point method is used to obtain the solution for OPF. However, due to the non-convex and nonlinearity of the OPF problem, it is challenging to meet the real-time requirement [2]. To expedite the solving process and obtain the OPF solution in near real-time, the convex relaxation [3] or linearization [4] forms of OPF problem are widely studied, e.g. direct-current optimal power flow (DC-OPF). However, such a simplified OPF problem may provide suboptimal or infeasible solutions.

Currently, since the OPF dataset for power systems can be simulated offline, it is possible to obtain an optimality and feasibility-enhanced OPF solution in real-time by shifting the computational burden from online optimization to offline training [5] [6]. Based on this idea, different data-driven OPF methods have been proposed, most of which are based on neural networks [7], such as deep neural networks (DNN) [8] [9], stacked extreme learning machines (SELM) [10], graph neural networks (GNN) [11], Graph Convolutional Neural Network (GCNN) [13], and convolutional neural networks (CNN) [12]. The core concept of these methods is to learn the mapping between the input (power demand) and the output (active power generation, reactive power generation, voltage magnitude, and voltage angle). Once the data-driven model is fully trained, the solution of OPF can be obtained in real-time by inputting the current power demand.

Even though the existing data-driven methods are proven to be efficient in obtaining OPF solutions in real-time, it remains a challenge for these methods to provide a feasible solution. To resolve this issue, several approaches have been proposed, which can be categorized into the penalty approach, projection approach, mapping approach, and implicit layer embedded approach.

For the penalty approach, it aims to add the physical constraint (equality or inequality constraint) to the loss function as the penalty term in order for the data-driven method to provide a feasible solution. In [14], the Karush-Kuhn-Tacker (KKT) conditions of OPF are embedded into the neural network to guide the training. In [15], the violation degrees for inequality and equality equations are designed and added to the neural network to capture the OPF constraints. Specifically, the max function and absolute function are used to measure the violation degrees for inequality and equality constraints, respectively. Although the penalty approach is able to generate OPF solutions with lower constraint violations, it still can not guarantee the feasibility of OPF solutions. Moreover, the penalty terms' weights are hard to select.

For the post-processing approach, it aims to obtain the preliminary solution of OPF (part or whole) from the neural network. Then, the final solution can be acquired through post-processing. In [16], two separate neural networks called voltage magnitude predictor (VMP) and voltage angle predictor (VAP) are used to produce the rough estimation of voltage magnitude and voltage angle. Then a two-step post-processing method is proposed to improve the feasibility of the rough estimation. Finally, the rest of the solutions are obtained based on the power flow equations and corrected estimation values. In [17], random forest is used to obtain an initial estimate of the OPF solution. This estimate is then used as a warm start for the traditional solver to obtain the final OPF solution. Since

The work was supported in part by the Research Grants Council of the Hong Kong SAR (HKU 27203723).

Yixiong Jia, Chenxi Wang, and Yi Wang are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong SAR, China, and are also with The University of Hong Kong Shenzhen Institute of Research and Innovation, Shenzhen, 518057, China (e-mail: jiayx@connect.hku.hk, cxwang@eee.hku.hk, yiwang@eee.hku.hk).

Yiqin Su is with the Department of Electrical Engineering, Tsinghua University, Beijing 100084, China (e-mail: suyg21@mails.tsinghua.edu.cn).

running the traditional solver to find the final solution can also be regarded as the additional post-processing step, we classify the method in [17] into the post-processing approach. In [17], an additional step to calculate the power flow equation is required, which means that accurate line parameters are required. Once the line parameters are not so accurate, the optimality and feasibility of the solution may be compromised. To improve the feasibility of the DC-OPF problem, in [18], the neural network is used to obtain a rough solution first. Then, the final feasible solution is obtained by solving a projection optimization problem. Despite the effectiveness of the approach, obtaining a real-time OPF solution remains difficult owing to the necessity for the post-processing step.

For the mapping approach, it aims to create a one-to-one mapping between the original feasible region and the unit ball in high dimension. Once the relaxed optimal solution in the unit ball is obtained, the feasible solution of the original OPF problem may be derived by applying the inverse mapping to the relaxed solution. To improve the feasibility of the DC-OPF problem, in [19], the gauge map is used to create the one-to-one mapping. Meanwhile, a feasibility module including inequality completion and equality completion is used to enforce the feasibility of the solution. To improve the feasibility of the AC-OPF problem, in [20], an invertible neural network is used to learn the one-to-one mapping called homeomorphic mapping between non-convex constraint and a unit ball in high dimension. Then, a bisection function is used to make sure the final solution satisfies the inequality constraints. The mapping approach can provide a feasible OPF solution because the solution is mapped into the feasible region. Nevertheless, a desirable feasible solution can be obtained only if the boundary of the non-convex feasible region is well mapped, which needs sufficient training data and requires the error of the mapping model to be zero. Unfortunately, the mapping error cannot be zero due to the neural network's prediction error; thus a feasible solution cannot be guaranteed.

For the implicit layer embedded approach, it aims to incorporate the post-processing step as an implicit layer within the neural network architecture and train the neural network in an end-to-end manner. Due to the implicit optimization layer embedded, the modified neural network can provide a feasible solution. To improve the feasibility of the DC-OPF problem, in [21], the projection step is implemented as the final layer of the deep neural network to guarantee the feasibility of the solution. To improve the feasibility of the AC-OPF problem, in [22], the estimation of the independent variable is provided by the neural network first, then the power flow equality function is implemented as an implicit layer to recover the rest dependent variable. To satisfy the inequality constraint, the simple unrolling gradient step is implemented. In [23], the primal-dual method is implemented as an implicit layer to enforce the output of the neural network to satisfy the inequality constraints during the training process. Compared to [22], [23] does not apply the primal-dual method in the inference step, which largely reduces the inference time. Despite the effectiveness of the implicit layer embedded approach, they still face several challenges. For [22] and [23], an unrolling inequality correction step and primal-dual method are used, respectively. However, these processes may compromise the equality feasibility.

To overcome the aforementioned challenges, we try to construct an OPF proxy that offers feasible solutions with acceptable time spent on problem-solving. In summary, we propose an OptNet-embedded data-driven OPF approach as a proxy for the OPF problem. The key contributions of this paper are summarized as follows,

- New model: We propose a data-driven OPF model by constructing a neural network architecture with OptNet layer embedded. This model only needs historical/simulated data but does not need network parameters for post-processing correction. Moreover, the proposed model serves as an OPF proxy to obtain a feasible solution by solving a linear approximation of the original OPF problem.
- New pruning method: Considering the high computational expense brought by OptNet layer embedding, we design a two-step pruning method to expedite the inference process of the proposed model. It is achieved by identifying and removing the unnecessary values and inequality constraints of the OptNet layer.

The rest of the paper is organized as follows: Section II gives the basic problem description of OPF and two questions this paper wants to answer. Section III describes the proposed three-stage neural network architecture and two-step pruning method to answer two questions. Section IV applies the proposed approach to the IEEE 4- and 14-bus systems. Section V draws conclusions and the future work.

## II. Problem Description

A standard OPF problem can be formulated as:

$$\min \sum_{i\in\Omega_G} \left(a_{2,i}\cdot(PG_i)^2 + a_{1,i}\cdot PG_i + a_{0,i}\right) \tag{1a}$$

$$P_{ij} = G_{ij}V_i^2 - V_iV_j\left(G_{ij}\cos\theta_{ij} + B_{ij}\sin\theta_{ij}\right), \forall i,j\in\Omega_L \tag{1b}$$

$$Q_{ij} = -B_{ij}V_i^2 - V_iV_j\left(G_{ij}\sin\theta_{ij} - B_{ij}\cos\theta_{ij}\right), \forall i,j\in\Omega_L \tag{1c}$$

$$PG_i - PD_i = \sum_{j\in i} P_{ij}, \forall i\in\Omega_B \tag{1d}$$

$$QG_i - QD_i = \sum_{j\in i} Q_{ij}, \forall i\in\Omega_B \tag{1e}$$

$$PG_i^{\min} \le PG_i \le PG_i^{\max}, \forall i\in\Omega_G \tag{1f}$$

$$QG_i^{\min} \le QG_i \le QG_i^{\max}, \forall i\in\Omega_G \tag{1g}$$

$$V_i^{\min} \le V_i \le V_i^{\max}, \forall i\in\Omega_B \tag{1h}$$

$$\sqrt{P_{ij}^2 + Q_{ij}^2} \le S_{ij}^{\max}, \forall i,j\in\Omega_L \tag{1i}$$

where (1a) is the objective function that aims to minimize the generation cost, (1b)-(1e) are the power flow

equation, (1f)-(1i) are the power, voltage, and power flow constraints, $a_{2,i}, a_{1,i}, a_{0,i}$ represent the cost coefficients of power generation at bus $i$, $\Omega_G, \Omega_B, \Omega_L$ are the generator set, bus set and branch set, respectively, $PG_i, QG_i$ are the active and reactive power generation at bus $i$, respectively, $PG_i^{\min}, PG_i^{\max}$ are minimum and maximum active power generation at generator $i$, $QG_i^{\min}, QG_i^{\max}$ are minimum and maximum reactive power generation at generator $i$, $PD_i, QD_i$ are the active and reactive power demand at bus $i$, respectively, $V_i, \theta_i$ are the voltage magnitude and voltage angle at bus $i$, $V_i^{\min}, V_i^{\max}$ are minimum and maximum voltage magnitude at bus $i$, $\theta_{ij} = \theta_i - \theta_j$ is the voltage angle difference between bus $i$ and bus $j$, $G_{ij}, B_{ij}$ are the conductance and susceptance of branch between bus $i$ and bus $j$, $P_{ij}, Q_{ij}$ are active and reactive power flow of branch between bus $i$ and bus $j$, $S_{ij}^{max}$ is the maximum power flow at branch between bus $i$ and bus $j$.

Data-driven OPF methods aim to learn the mapping from the input $\boldsymbol{x}^k$ to the output $\boldsymbol{y}^k$ based on the dataset generated by solving (1). This process could be formulated as (2).

$$\min_{\boldsymbol{w}} Loss\left(\boldsymbol{X}, \boldsymbol{Y} \middle| \boldsymbol{w}\right) = \frac{1}{n_{\text{train}}} \sum_{k=1}^{n_{\text{train}}} \mathcal{L}\left(\boldsymbol{y}^k, f_{NN}\left(\boldsymbol{x}^k \middle| \boldsymbol{w}\right)\right) \tag{2}$$

where $f_{NN}\left(\cdot \middle| \boldsymbol{w}\right) : \mathcal{R}^{2n} \rightarrow \mathcal{R}^{2m+2n-2}$ represents the mapping with parameter $\boldsymbol{w}$, $n$ denotes the number of bus, $m$ denotes the number of generator of the system, $D = \{(\boldsymbol{X}, \boldsymbol{Y})\} = \left\{\left(\boldsymbol{x}^k, \boldsymbol{y}^k\right)\right\}_{k=1}^{n_{\text{train}}}$ is the training dataset, $n_{\text{train}}$ represents the number of training sample, $\boldsymbol{x}^k = \left(\boldsymbol{PD}^k, \boldsymbol{QD}^k\right)$ represents the $k_{th}$ input, $\boldsymbol{y}^k = \left(\boldsymbol{PG}^k, \boldsymbol{QG}^k, \boldsymbol{V}^k, \boldsymbol{\theta}^k\right)$ represents the $k_{th}$ OPF solution, $\mathcal{L}\left(\cdot, \cdot\right)$ denotes the loss function. Since the learning objctive is to obtain an accurate OPF solution, the mean square error (MSE) is used as the loss function, which can be formulated as:

$$\mathcal{L}\left(\boldsymbol{y}^k, f_{NN}\left(\boldsymbol{x}^k \middle| \boldsymbol{w}\right)\right) = \left(\boldsymbol{y}^k - f_{NN}\left(\boldsymbol{x}^k \middle| \boldsymbol{w}\right)\right)^2 \tag{3}$$

The aim of this work is to combine the neural networks and optimization layer to provide a feasible solution for the AC-OPF problem. To achieve this goal, two questions need to be answered:

1) How to construct a neural network architecture as an OPF proxy to provide a feasible solution?
2) How to expedite the model inference process and minimize model accuracy loss?

In the following page, the first question will be answered in subsection III-A by designing a three-stage neural network architecture, and the second question will be answered in subsection III-C by removing unnecessary values and inequality constraints of the OptNet layer.

## III. Methodology

In this section, a three-stage neural network architecture is designed to serve as an OPF proxy. Specifically, the OptNet layer is embedded into the neural network architecture to provide a feasible solution. In addition, considering the OptNet layer could slow the inference step, a two-step pruning method is also proposed.

### A. Neural Network Architecture Design

In this subsection, we describe the architecture of the proposed model. The OptNet layer [24] has the ability to learn and provide a feasible solution to an optimization problem (for example, a Quadratic Programming problem). With the help of this layer, the architecture of the proposed model is designed in Fig 1. The $\boldsymbol{w}_1, \boldsymbol{w}_3$ are trainable parameters of neural networks in stage 1 and stage 3, respectively.

According to Fig 1, one may observe that the architecture design consists of three stages. The first stage is designed to project the input to a high-dimensional space. Then, the second stage incorporates an OptNet layer to solve the core optimization problem. Finally, the third stage maps the high-dimensional output obtained from the OptNet layer to the OPF solution.

Typically, for stage 2, the OptNet layer [24] is defined to solve a quadratic programming problem, which can be formulated as:

$$\boldsymbol{z}_2 = \arg\min_{\boldsymbol{z}} \frac{1}{2}\boldsymbol{z}^T \cdot \boldsymbol{Q} \cdot \boldsymbol{z} + \boldsymbol{c}^T \cdot \boldsymbol{z} \tag{4a}$$

$$\text{s.t.} \quad \boldsymbol{A} \cdot \boldsymbol{z} = \boldsymbol{b}\left(\boldsymbol{z}_1\right), \quad \boldsymbol{E} \cdot \boldsymbol{z} \leq \boldsymbol{h} \tag{4b}$$

where $\boldsymbol{z} \in \mathcal{R}^{n_Q \times 1}$ is the optimization variable, $\boldsymbol{z}_2$ is the output of OptNet layer, $\boldsymbol{Q} \in \mathcal{R}^{n_Q \times n_Q}, \boldsymbol{c} \in \mathcal{R}^{n_Q \times 1}, \boldsymbol{A} \in \mathcal{R}^{n_{\text{eq}} \times n_Q}, \boldsymbol{E} \in \mathcal{R}^{n_{\text{ineq}} \times n_Q}, \boldsymbol{h} \in \mathcal{R}^{n_{\text{ineq}} \times 1}$ are parameters will be learned from the dataset, $\boldsymbol{b}\left(\boldsymbol{z}_1\right) \in \mathcal{R}^{n_{\text{eq}} \times 1}$ is a parameter related to the previous layer output $\boldsymbol{z}_1$, $n_Q$ represents the dimension of lifted space, $n_{\text{eq}}$ represents the dimension of the equality constraints, and $n_{\text{ineq}}$ represents the dimension of the inequality constraints.

The idea behind this architecture and why the architecture can serve as an OPF proxy to provide a feasible solution is easy to understand. Because of the non-linearity and non-convexity of (1), solving the standard OPF problem is challenging. To tackle this issue, various linear approximations of OPF have been proposed. Among the approximation methods, the most famous one is DC-OPF, which can be formulated as a quadratic programming problem. Also, comparing (1) with (4), one may observe that once the power flow equation is transformed into a linear one, (1) becomes a quadratic programming problem. In addition, [25] and [26] suggest that the power flow equation approximates linear when the state is lifted to a high-dimensional space.

Thus, following these two ideas, Stage 1 is utilized to lift the dimension of the input. Then, the OPF can be considered as a quadratic programming problem and can be solved in Stage 2, where the input $\boldsymbol{b}\left(\boldsymbol{z}_1\right)$ is equal to the output of Stage 1. Since the dimension is lifted, the output from Stage 2 differs from the original OPF solution. Besides, the estimated solution of (1) is coupled in the output of Stage 2, and the coupled one exactly satisfies

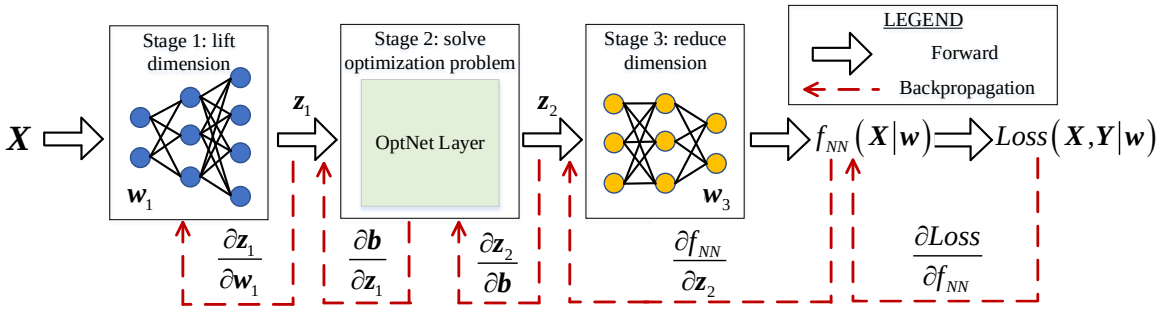


Fig. 1. Architecture of the proposed data-driven OPF model

the constraints in high dimension. To decouple the output of Stage 2 and obtain the final OPF solution, in Stage 3, we use another neural network. In summary, the proposed three-stage architecture can be considered as a proxy of the OPF problem (1). Once the model is fully trained, the forward pass can be considered to solve the linearization OPF problem. In this way, we do not need to use correction steps in our model and a feasible solution can be obtained.

## B. Backpropagation

To update the trainable parameters of the proposed model, we need to calculate the gradient of the loss function with respect to the parameters. To elaborate on this process, we take two trainable parameters as examples. Typically, the gradient can be expressed as (5). Note that the brackets with its variables are omitted on the right-hand side in (5).

$$\nabla_{\boldsymbol{w}_3} Loss\left(\boldsymbol{X},\boldsymbol{Y}|\,\boldsymbol{w}\right)=\frac{\partial Loss}{\partial f_{NN}}\cdot\frac{\partial f_{NN}}{\partial \boldsymbol{w}_3} \tag{5a}$$

$$\nabla_{\boldsymbol{w}_1} Loss\left(\boldsymbol{X},\boldsymbol{Y}|\,\boldsymbol{w}\right)=\frac{\partial Loss}{\partial f_{NN}}\frac{\partial f_{NN}}{\partial \boldsymbol{z}_2}\frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{b}}\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{z}_1}\frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{w}_1} \tag{5b}$$

In (5), the term $\frac{\partial Loss}{\partial f_{NN}}$ can be analytically obtained based on (2) and (3), $\frac{\partial f_{NN}}{\partial \boldsymbol{w}_3}$, $\frac{\partial f_{NN}}{\partial \boldsymbol{z}_2}$ and $\frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{w}_1}$ are widely used to update the parameters of neural network and can be calculated easily. Thus, the key step to calculate (5b) is to obtain the $\frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{b}}$ and $\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{z}_1}$. To obtain $\frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{b}}$, we follow the derivation process in [24].

The Lagrangian of (4) can be formulated as (6),

$$L\left(\boldsymbol{z},\nu,\lambda\right)=\frac{1}{2}\boldsymbol{z}^T\boldsymbol{Q}\boldsymbol{z}+\boldsymbol{c}^T\boldsymbol{z}+\boldsymbol{\nu}^T\left(\boldsymbol{A}\boldsymbol{z}-\boldsymbol{b}\left(\boldsymbol{z}_i\right)\right)+\boldsymbol{\lambda}^T\left(\boldsymbol{E}\boldsymbol{z}-\boldsymbol{h}\right) \tag{6}$$

where $\boldsymbol{\nu},\boldsymbol{\lambda}\geq 0$ are Lagrangian multipliers.

By differentiating the KKT conditions of (6), the linear matrix form can be obtained as (7),

$$\begin{bmatrix}\boldsymbol{Q} & \boldsymbol{E}^T & \boldsymbol{A}^T\\ \boldsymbol{G} & \boldsymbol{H} & 0\\ \boldsymbol{A} & 0 & 0\end{bmatrix}\begin{bmatrix}d\boldsymbol{z}_2\\ d\boldsymbol{\lambda}^*\\ d\boldsymbol{\nu}^*\end{bmatrix}=\begin{bmatrix}0\\0\\d\boldsymbol{b}\end{bmatrix} \tag{7}$$

where $\boldsymbol{z}_2$, $\boldsymbol{\lambda}^*$ and $\boldsymbol{\nu}^*$ are optimal values for primal and dual variable, $\boldsymbol{G}=diag\left(\boldsymbol{\lambda}^*\right)\boldsymbol{E}$, $\boldsymbol{H}=diag\left(\boldsymbol{E}\cdot\boldsymbol{z}_2-\boldsymbol{h}\right)$, $diag\left(\cdot\right)$ is an operator extracts the vector $(\cdot)$ to the diagonal matrix. Thus, the $\frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{b}}$ can be obtained by solving the (7).

For $\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{z}_1}$, to simplify the architecture, we set $\boldsymbol{z}_1=\boldsymbol{b}$. That means, $\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{z}_1}=\boldsymbol{I}$ in this paper. $\boldsymbol{I}$ represents the identity matrix. Note that, the other parameters like $\frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{Q}}$ can be obtained in a similar way like (7). And the gradient $\nabla_{\boldsymbol{Q}} Loss$ can be obtained like (5a) or (5b). Once the gradient of each trainable parameter is known, then the parameters can be updated using gradient-type methods.

## C. Model Pruning

In this subsection, we describe the proposed two-step pruning method. First, we randomly generate different testing samples and compute the total inference time spent in Stages 1 and 3 and Stage 2, as shown in Fig. 2 abd Fig. 3, respectively. According to Fig. 2 and 3, one may observe that almost 99% of inference time is spent in Stage 2. Thus, reducing the time spent on solving optimization problem (4) is crucial to expedite the inference process. In this regard, we proposed a two-step pruning method to expedite the inference process with a little loss of accuracy. The first step utilizes historical data to prune the inactive inequality constraints away. Following the idea of the neural networks pruning method, the second step removes the unnecessary values of parameters in problem (4).

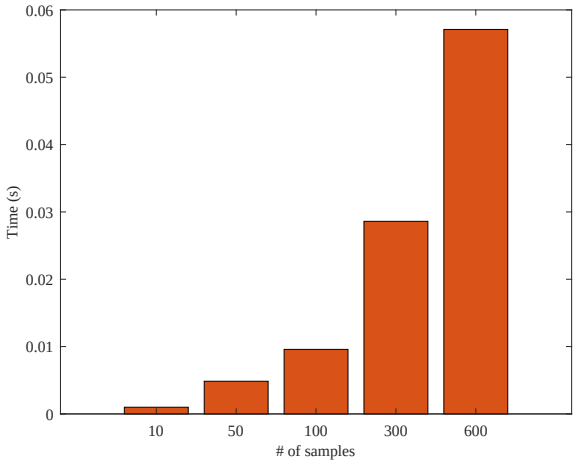


Fig. 2. Time spent in Stages 1 and 3

1) Pruning away the inactive inequality constraints: For the optimization problem (4), the $n_{\text{ineq}}$ is pre-defined to train the model. However, similar to the original OPF

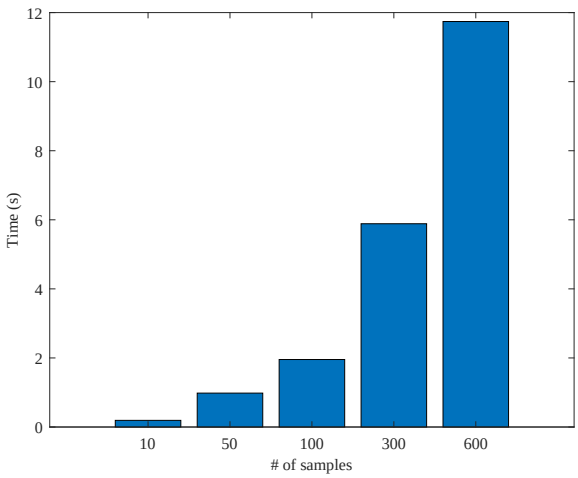


Fig. 3. Time spent in Stages 2

problem [27], many of the inequality constraints in (4) are inactive and do not affect the optimum of the problem. That means once the inactive inequality constraints are removed, the inference time for the optimization problem (4) will be reduced. In this regard, we define (8) to measure the inequality distance and use (9) to find the inactive inequality constraint set.

$$d_{l,k} = \hat{\boldsymbol{h}}^{(l)} - \hat{\boldsymbol{E}}^{(l)} \cdot \boldsymbol{z}_2^k \tag{8}$$

where $\hat{\boldsymbol{E}}^{(\boldsymbol{l})}$ represents the $l_{th}$ row of the fully trained parameter $\hat{\boldsymbol{E}}$, $\hat{\boldsymbol{h}}^{(l)}$ represents the $l_{th}$ row of the fully trained parameter $\hat{\boldsymbol{h}}$, $\boldsymbol{z}_2^k$ represents the optimal solution of (4) by inputting the $k_{th}$ training sample, $l \in \{1, \cdots, n_{\text{ineq}}\}$, $k \in \{1, \cdots, n_{\text{train}}\}$.

$$\begin{aligned} l &= \max_{l \in \chi} \|\boldsymbol{d}_{l,*}\|_2 \\ \text{s.t.} \quad & d_{l,k} > \varepsilon, \quad \forall k \in \{1, \cdots, n_{\text{train}}\} \end{aligned} \tag{9}$$

where $\chi$ represents the active set, $\varepsilon$ is used as a threshold, $\boldsymbol{d}_{l,*} = \begin{bmatrix} d_{l,1} & \cdots & d_{l,n_{\text{train}}} \end{bmatrix}$.

The inactive set $\phi$ of inequality constraint can be constructed iteratively by solving (9). For example, once a new index $l$ is found to be inactive, then this index will be removed from $\chi$, and this index will be added to the $\phi$. The final $\phi$ can be constructed until no inactive inequality constraint index can be found.

The reason we use an iterative method to construct the inactive inequality constraint set is easy to understand. Since removing one inequality constraint may affect the other, the iterative approach is a good way to find all the inactive inequality constraints. It should be noted that the inactive constraints set may be changed when the range of power demand varies. To accommodate almost every scenario and ensure the pruning method can be used in the practical system, we sample the power demand from 0.85 to 1.45 in our training dataset.

With the help of $\phi$, the inequality constraint in (4b) can be equivalently simplified as (10),

$$\hat{\boldsymbol{E}}_{\boldsymbol{\phi}} \cdot \boldsymbol{z} \leq \hat{\boldsymbol{h}}_{\boldsymbol{\phi}} \tag{10}$$

where $\hat{\boldsymbol{E}}_{\boldsymbol{\phi}}$ and $\hat{\boldsymbol{h}}_{\boldsymbol{\phi}}$ are constructed by extracting selected rows from $\boldsymbol{E}$ and $\boldsymbol{h}$, the selected rows are in $\phi$.

2) Pruning away the unnecessary values of parameter: Similar to the deep neural network [28], the values in the fully trained parameters in (4) may be redundant and may be over-fitted in the training dataset. That means once the redundant values are dropped, the performance of the proposed model may not be affected, and the computational expense to solve the optimization problem (4) will be reduced because of the sparsity. Following this idea, the pruning process of $\hat{\boldsymbol{Q}}$ can be visualized in Fig 4, where $Tr$ represents the pruning threshold; the position with a red dashed circle represents the unnecessary values of the parameters. The white blocks in Fig. 4 represent zero values, and the colour blocks represent non-zero values. Note that, the other parameters like $\hat{\boldsymbol{c}}$ also need to be pruned. Since the matrix $Q$ is decomposed to $U^TU+eps$ in the OptNet layer, the estimation of $Q$ will always be symmetric in our method.

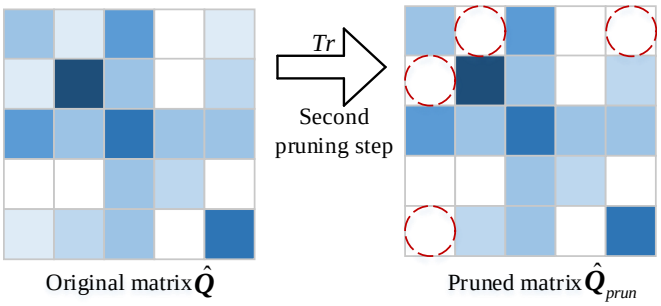


Fig. 4. Pruning process of $\hat{\boldsymbol{Q}}$

With the help of two-step's pruning, the OptNet layer for the stage 2 can be reformulated as (11).

$$\boldsymbol{z}_2 = \arg\min_{\boldsymbol{z}} \frac{1}{2}\boldsymbol{z}^T \cdot \hat{\boldsymbol{Q}}_{prun} \cdot \boldsymbol{z} + \hat{\boldsymbol{c}}_{prun}^T \cdot \boldsymbol{z} \tag{11a}$$

$$\text{s.t.} \quad \hat{\boldsymbol{A}}_{prun} \cdot \boldsymbol{z} = \hat{\boldsymbol{b}}_{prun}(\boldsymbol{z}_1), \quad \hat{\boldsymbol{E}}_{\phi} \cdot \boldsymbol{z} \leq \hat{\boldsymbol{h}}_{\phi} \tag{11b}$$

where the subscript "prun" represents the pruned matrix or vector.

Compared to (4), the redundant constraints and the values are all removed. Hence, the computational efficiency of the inference step (especially for stage 2 in the proposed method) is significantly enhanced.

## IV. Case Studies

In this section, to visualize the performance of the proposed method, a 4-bus system is utilized first. Then, OPF results comparison between the proposed method and other baseline methods is shown based on a modified 14-bus system.

### A. Simulation Settings

Test Case: In order to demonstrate the effectiveness of the proposed method, 4- and 14-bus systems are utilized. The detail of the 4- and 14-bus system are shown in Fig 5 and 6, respectively.

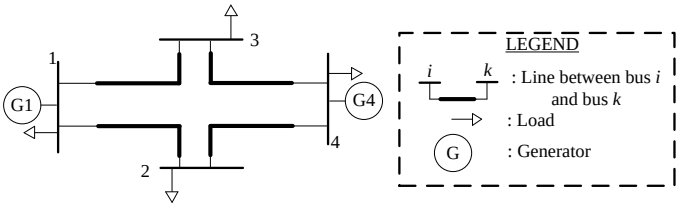


Fig. 5. The 4-bus system

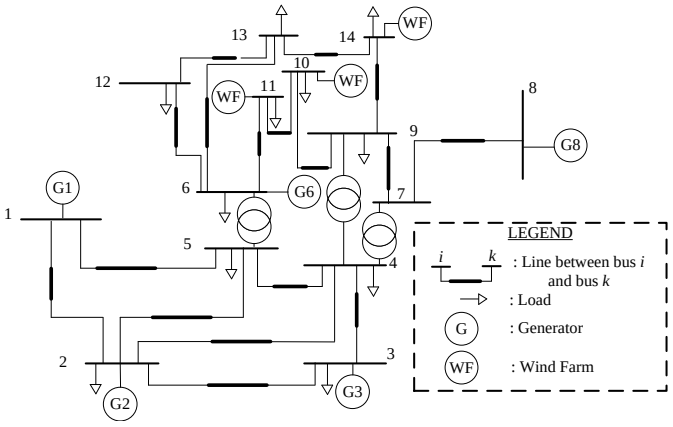


Fig. 6. The modified 14-bus system

Data Generation: For the 4- and 14-bus systems, the load dataset is generated based on the standard value and the uniform distribution shown as (12) [29]. The $\alpha$ is used to capture the load variation, and $\beta$ is used to capture the weather influence on the load. Note that, when $\alpha_i \sim U[1.15, 1.45]$ we call it a heavy load scenario, and when $\alpha_i \sim U[0.85, 1.15]$ we call it a light load scenario. Note that we sample the power demand from 0.85 to 1.45 to contain almost every scenario. If the load demand is outside the sample range, the optimality, and feasibility of the solution may be affected. In addition, the power output of each wind farm is generated by using the parameters and the Weibull distribution in [30], which can be formulated as (13). The $PW_i$ is the wind farm power output at bus $i$, $v_i$ is the wind speed of the wind farm that is connected to bus $i$, $PW_{i,r}$ is the rate capacity of the wind farm that is connected to bus $i$, and $v_{i,in}, v_{i,r}, v_{i,out}$ are the cut-in, rate, and cut-out wind speed of the wind farm that is connected to bus $i$. Then, the OPF dataset is constructed by using PyPower. Specifically, for the 4-bus system, 5000 samples are generated for training. For the 14-bus system, 3000 samples are generated in the heavy load scenario and light load scenario, respectively, where 80% samples are used for training and the other 20% samples are used for testing.

$$PD_i = (\alpha_i + \beta_i) \cdot PD_i^{standard}, \forall i \in \Omega_B \tag{12a}$$

$$QD_i = (\alpha_i + \beta_i) \cdot QD_i^{standard}, \forall i \in \Omega_B \tag{12b}$$

$$\alpha_i \sim U[0.85, 1.45], \beta_i \sim U[-0.025, 0.025] \tag{12c}$$

$$PW_i = \begin{cases} 0, & if \quad 0 \le v_i \le v_{i,in} \\ PW_{i,r} \cdot \frac{v_i - v_{i,in}}{v_{i,out} - v_{i,in}}, & if \quad v_{i,in} \le v_i \le v_{i,r} \\ PW_{i,r}, & if \quad v_{i,r} \le v_i \le v_{i,out} \\ 0, & if \quad v_{i,out} \le v_i \end{cases} \tag{13a}$$

Comparison Methods: the performance of the proposed method is compared with five other methods, including, 1) Optimizer, which uses PyPower to obtain the OPF solution; 2) neural network (NN), which trains a neural network on the dataset to obtain the OPF solution; 3) neural network-penalty (NN-P), which adds the penalty term based on the physical constraint to the loss function for the neural network training; 4) DC3, which applies equality completion and unrolling gradient steps to recover the feasibility of the solution provided by the neural network; 5) neural network-OptNet (NN-OptNet), which adopts OptNet layer in the neural network to provide feasible OPF solution; 6) neural network-OptNet-Pruning (NN-OptNet-Pr), which applies the proposed two-step pruning method to speed up the inference process of NN-OptNet. The illustration of comparison methods is shown as Tab. I

TABLE I
Illustration of Comparison Methods

| Methods Name | NN | NN-P | DC3 | NN-OptNet | NN-OptNet-Pr |
|---|---|---|---|---|---|
| Reference | [5] | [15] | [22] | Proposed | Proposed |
| Type | purely data-driven approach | penalty approach | implicit layer embedded approach | implicit layer embedded approach | implicit layer embedded approach |

Model Structure: To reduce the inference time of NN-OptNet, only one layer is adopted in stage 1 and stage 3, respectively. For the 14-bus system, $n_Q$ is set to 60, and $n_{\text{eq}}, n_{\text{ineq}}$ are assigned as 20. After the pruning, the $n_{ineq}$ is assigned as 12. The details of the hyper-parameters selection are analyzed in subsection IV-D.

Evaluation Metrics: To evaluate the performance of the proposed method, four criteria are used, where $n_{\text{test}}$ represents the number of testing sample. 1) Optimality: The mean relative objective function error is used to evaluate the optimality of the method, which can be formulated as (14). For (14), $c(\cdot)$ represents the objective function shown as (1a), $\widehat{\boldsymbol{PG}}$ represents the active power output from the data-driven OPF method; 2) Equality Feasibility: The mean power flow equation error is used to evaluate the equality feasibility of the method, which can be formulated as (15). For (15), $\widehat{\boldsymbol{PG}}$, $\widehat{\boldsymbol{QG}}$, $\widehat{\boldsymbol{V}}$ and $\widehat{\boldsymbol{\theta}}$ represents the OPF solution from the data-driven method, $PF(\cdot,\cdot,\cdot,\cdot)$ represents the power flow equation shown as (1b) to (1e); 3) Inequality Feasibility: The maximum violation rate is used to evaluate the inequality feasibility of the method, which can be formulated as (16). For (16), the function $g(\cdot)$ is used to obtain the violation rate for the variable $(\cdot)$, $g_s(\cdot)$ is step function. 4) Time Complexity: The total inference time in the testing dataset is used to evaluate the time complexity of the method, which can

be formulated as (17). The function $time(\cdot)$ is used to calculate the inference time. Note that, for a traditional solver, the sum of the solving time for the total testing samples is utilized to evaluate the time complexity.

$$\frac{1}{n_{\text{test}}}\sum_{k=1}^{n_{\text{test}}}\left(\frac{\left|c\left(\widehat{\boldsymbol{PG}}^{k}\right)-c\left(\boldsymbol{PG}^{\boldsymbol{k}}\right)\right|}{c\left(\boldsymbol{PG}^{\boldsymbol{k}}\right)}\right) \tag{14}$$

$$\frac{1}{n_{\text{test}}}\sum_{k=1}^{n_{\text{test}}}\left|PF\left(\widehat{\boldsymbol{PG}}^{k},\widehat{\boldsymbol{QG}}^{k},\hat{\boldsymbol{V}}^{k},\hat{\boldsymbol{\theta}}^{k}\right)\right| \tag{15}$$

$$\frac{1}{\{n,m\}\cdot n_{\text{test}}}\cdot\max\left\{g\left(\widehat{\boldsymbol{PG}}\right),g\left(\widehat{\boldsymbol{QG}}\right),g\left(\hat{\boldsymbol{V}}\right),0\right\} \tag{16a}$$

$$g\left(\hat{\boldsymbol{M}}\right)=\sum_{i\in\{\Omega_B,\Omega_G\}}\sum_{k=1}^{n_{\text{test}}}\left[\begin{array}{l}g_s\left(\hat{\boldsymbol{M}}_i^k-M_i^{\max}\right)\\+g_s\left(M_i^{\min}-\hat{\boldsymbol{M}}_i^k\right)\end{array}\right] \tag{16b}$$

$$\sum_{k=1}^{n_{\text{test}}} time\left(f_{NN}\left(\boldsymbol{x}^k\middle|\hat{\boldsymbol{w}}\right)\right) \tag{17}$$

## B. Case Study on 4-bus System

To visually demonstrate the optimality of the proposed method for the 4-bus system, we fix the active power demand at buses 3 and 4, as well as the reactive power demand at buses 1 to 4. For buses 1 and 2, 200 active power demands are generated using (12), respectively. Then 40000 testing cases are generated by corresponding to each other. By using the optimizer and the proposed method, the objective function value comparison for the 40000 testing cases is shown in Fig 7. The horizontal and vertical axes represent the active power demand (unit: MW) at buses 1 and 2, respectively. Note that, the different color represents different objective function value (unit: \$/hr), with more similar colors indicating closer objective function values. Compare the top figure and the bottom figure in Fig 7, one may observe that the NN-OptNet is capable of generating accurate OPF solutions for most cases and only for small cases on the right-hand side, the NN-OptNet may provide conservative OPF solutions resulting in lower power generation estimation compared to the optimizer.

In addition to studying the optimality of the proposed method, the inequality feasibility is also studied, which is demonstrated in Fig 8. The horizontal and vertical axes represent the estimated power generation (unit: MW) at buses 1 and 4, respectively. Note that, the blue circle and orange cross represent the power generation obtained from optimizer and NN-OptNet, respectively. Overall, the estimation results obtained from NN-OptNet are roughly similar to those obtained from the Optimizer. This is confirmed by taking a close look at the left area with a red dash square. In addition, all the power generation results obtained from NN-OptNet lay in the feasible region, which can be verified from the right area with a red dash square. All of these findings can confirm that the proposed method can be considered as a proxy of the original OPF problem (1).

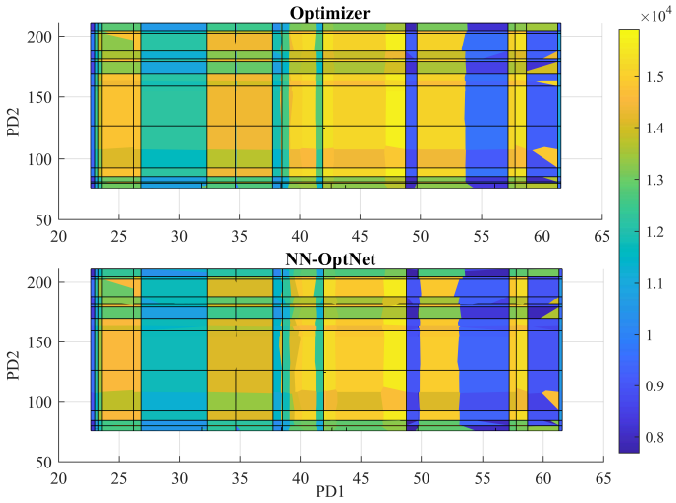


Fig. 7. Objective function value comparison between Optimizer and NN-OptNet

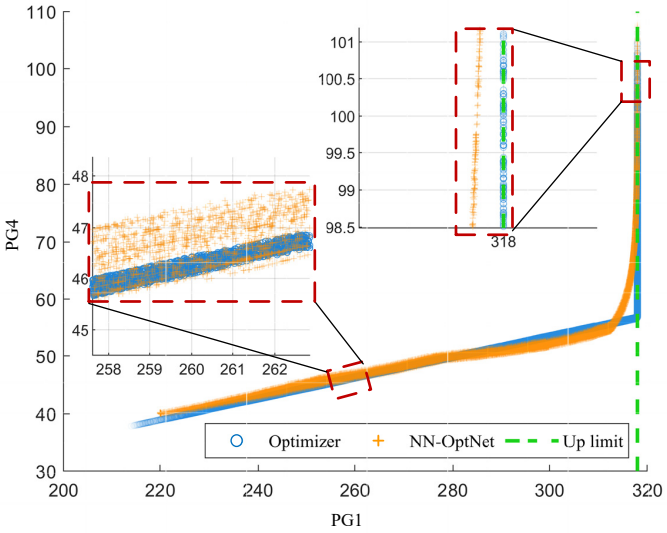


Fig. 8. Power generation comparison between Optimizer and NN-OptNet

## C. Case Study on 14-bus System

The performance comparison between various methods in the heavy load scenario and light load scenario is shown in Table II and Table III, respectively.

The second column in Table II and Table III compare the Optimalty for different methods. One may observe that the NN-OptNet achieves the smallest value compared to the other three methods no matter in the heavy load scenario or light load scenario. It means that the average difference between the actual objective function value and the estimated objective function value is the smallest, which confirms the effectiveness of the proposed method. Note that, the Optimality value for the NN-P is larger than the NN in the heavy load scenario. It is because when we use a large penalty value for NN-P, the violation rate of NN-P will decrease, however, the large penalty value will also have a negative impact on the Optimality.

Comparing the NN-OptNet with NN-OptNet-Pr, one may observe that a suitable pruning will boost the performance of the NN-OptNet method in a light load scenario or heavy load scenario. However, it is a challenge to select the proper threshold to balance the model performance in heavy and light scenarios.

The third column in Table II and Table III compare the Equality Feasibility for different methods. One may observe that the proposed method shows the best equality feasibility maintained compared to the other three methods. For example, the equality feasibility for the NN-OptNet is 1.10E-02, while this value reaches 3.51E-02 in DC-3. Comparing the NN-OptNet with NN-OptNet-Pr, the findings are similar to the Optimalty.

The fourth column in Table II and Table III compare the Inequality Feasibility for different methods. It should be noted that both the NN-OptNet and NN-OptNet-Pr can maintain a zero violation rate. For the DC-3, the inequality feasibility can be recovered to zero by setting more steps in the inequality correction step, however, the time cost will also be increased and the equality feasibility may also be affected. Similar to DC-3, the inequality feasibility of NN-P can also be recovered to zero, however, the equality feasibility and optimality may be affected.

The fifth column in Table II and Table III compare the Time Complexity for different methods. Both NN and NN-P show excellent computational efficiency. However, they may provide infeasible and sub-optimal OPF solutions. Compared to the other three methods, the NN-OptNet needs more time in the inference process because the optimization layer is embedded. The NN-OptNet-Pr needs less time to obtain the OPF solution compared to the NN-OptNet but the performance may also be compromised.

TABLE II
Performance Comparison in Heavy Load Scenario

| | Optimality | Equality Feasibility | Inequality Violation Rate | Time |
|---|---|---|---|---|
| Optimizer | – | - | – | 143.01s |
| NN | 5.68E-03 | 7.99E-02 | 23.30% | 0.072s |
| NN-P | 1.14E-02 | 4.65E-02 | 7.16% | 0.075s |
| DC-3 | 3.62E-03 | 3.51E-02 | 2.54% | 3.22s |
| NN-OptNet | 2.22E-03 | 1.10E-02 | 0.00% | 11.41s |
| NN-OptNet-Pr | 4.95E-03 | 1.06E-02 | 0.00% | 6.82 s |

TABLE III
Performance Comparison in Light Load Scenario

| | Optimality | Equality Feasibility | Inequality Violation Rate | Time |
|---|---|---|---|---|
| Optimizer | – | - | – | 144.40s |
| NN | 9.37E-03 | 8.43E-02 | 13.04% | 0.071s |
| NN-P | 7.43E-03 | 4.44E-02 | 6.33% | 0.076s |
| DC-3 | 6.95E-03 | 3.62E-02 | 3.00% | 3.21s |
| NN-OptNet | 3.90E-03 | 8.62E-03 | 0.00% | 11.43s |
| NN-OptNet-Pr | 6.65E-04 | 8.90E-03 | 0.00% | 6.56 s |

## D. Hyper-parameter Selection

Except for the hyper-parameters like learning rate, for the NN-OptNet, the hyper-parameters also include the number of layers, the dimension of lifted space $n_Q$, the dimension of the equality function $n_{\text{eq}}$, and the dimension of inequality function $n_{\text{ineq}}$ in the optimization layers. To ensure the minimum inference time, the number of layers is set to 1 in both stage 1 and stage 3. To determine the other hyper-parameter, we use Fig 9 to show the time complexity with different $n_{\text{eq}}/n_{\text{ineq}}$ and $n_Q$. Note that, we set $n_{eq} = n_{ineq}$ in the figure.

The x and y axes represent the dimension of lifted space $n_Q$ and the dimension of the equality/inequality function $n_{\text{eq}}/n_{\text{ineq}}$, respectively. The z axes represent the inference time spent for the NN-OptNet method. Note that, the problem (4) may be unsolvable when $n_{\text{eq}}/n_{\text{ineq}} > n_Q$. Thus, we set solving time to zero in Fig (9) when the problem is unsolvable. There is no doubt that the higher dimension may increase model accuracy, but also increase time complexity. Thus, to balance the model performance and the inference speed, we select $n_Q = 60$, $n_{\text{eq}} = 20, n_{\text{ineq}} = 20$ according to Fig (9).

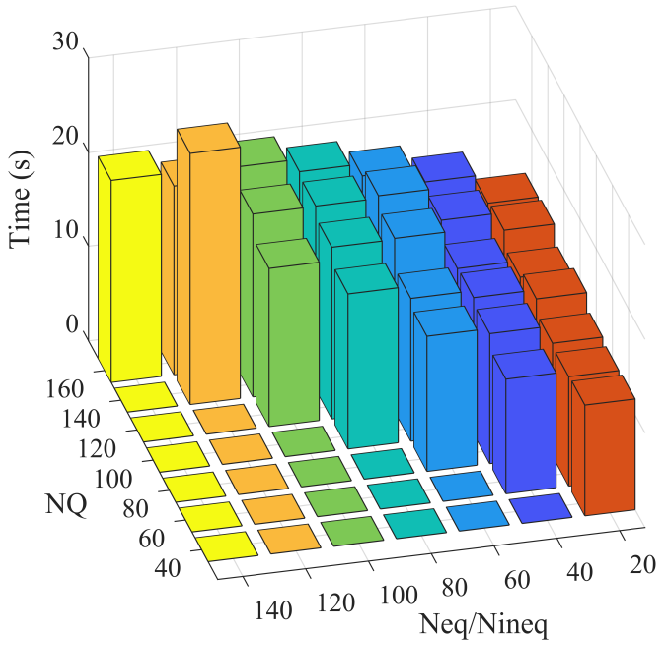


Fig. 9. Inference time comparison with different hyper-parameters

## E. Discussion

Since the OptNet layer is embedded in the proposed architecture, our method will face the scalability issue for a large power grid. Meanwhile, even though we can get a feasible OPF solution from the simulation perspective, the feasibility cannot be guaranteed from the theoretical perspective.

In addition, we would like to compare the proposed method and the convex-relaxation layer embedded approach in our future work.

## V. Conclusions

This paper proposes to represent the OPF problem by a three-stage neural network architecture with the OptNet layer embedded. The forward pass of the model is equivalent to solving a linearization OPF problem. Thus, the correction steps are not required for the proposed approach. Then, a two-step pruning method is proposed to identify and remove the unnecessary inequality constraints and values to expedite the inference step. Therefore, the computational burden of the proposed model can be reduced. While we do not claim that the feasibility can be guaranteed by using the proposed approach, numerical experiments based on the IEEE 4- and 14-bus systems show that the proposed approach can achieve positive results in terms of feasibility and optimality compared to the other three methods.

The quadratic programming problem is used in the OptNet layer. However, QP is not a perfect approximation of the OPF problem. In our future work, we would like to construct or find an optimization layer to provide better OPF solutions with fewer time costs in the inference step. In addition, we would like to implement our approach to the practical power grid to examine its practical value.

## References


[1] N. Guha, et al. "Machine learning for AC optimal power flow." arXiv preprint arXiv:1910.08842 (2019).

[2] Y. Tang, K. Dvijotham, and S. Low,"Real-time optimal power flow," IEEE Trans. Smart Grid, vol. 8, no. 6, pp. 2963–2973, Nov. 2017.

[3] D. K. Molzahn, "Incorporating Squirrel-Cage Induction Machine Models in Convex Relaxations of OPF Problems," IEEE Trans. Power Syst., vol. 32, no. 6, pp. 4972-4974, Nov. 2017.

[4] A. Castillo, P. Lipka, J. -P. Watson, S. S. Oren and R. P. O'Neill, "A successive linear programming approach to solving the iv-acopf," IEEE Trans. Power Syst., vol. 31, no. 4, pp. 2752-2763, July 2016.

[5] T. Falconer and L. Mones, "Leveraging Power Grid Topology in Machine Learning Assisted Optimal Power Flow," IEEE Trans. Power Syst., vol. 38, no. 3, pp. 2234-2246, May 2023.

[6] Y. Chen, S. Lakshminarayana, C. Maple and H. V. Poor, "A Meta-Learning Approach to the Optimal Power Flow Problem Under Topology Reconfigurations," IEEE Open Access J. Power Energy, vol. 9, pp. 109-120, 2022.

[7] S. Liu, C. Wu, and H. Zhu, "Topology-aware Graph Neural Networks for Learning Feasible and Adaptive AC-OPF Solutions," IEEE Trans. Power Syst., pp. 1–11, 2022.

[8] X. Pan, M. Chen, T. Zhao and S. H. Low, "DeepOPF: A Feasibility-Optimized Deep Neural Network Approach for AC Optimal Power Flow Problems," IEEE Syst. J., vol. 17, no. 1, pp. 673-683, March 2023.

[9] A. S. Zamzam and K. Baker, "Learning Optimal Solutions for Extremely Fast AC Optimal Power Flow," 2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), Tempe, AZ, USA, 2020, pp. 1-6.

[10] X. Lei, Z. Yang, J. Yu, J. Zhao, Q. Gao and H. Yu, "Data-Driven Optimal Power Flow: A Physics-Informed Machine Learning Approach," IEEE Trans. Power Syst., vol. 36, no. 1, pp. 346-354, Jan. 2021.

[11] T. Pham and X. Li, "Reduced Optimal Power Flow Using Graph Neural Network," 2022 North American Power Symposium (NAPS), Salt Lake City, UT, USA, 2022, pp. 1-6.

[12] K. Yang, W. Gao and R. Fan, "Optimal Power Flow Estimation Using One-Dimensional Convolutional Neural Network," 2021 North American Power Symposium (NAPS), College Station, TX, USA, 2021, pp. 1-6.

[13] M. Gao, J. Yu, Z. Yang and J. Zhao, "A Physics-Guided Graph Convolution Neural Network for Optimal Power Flow," IEEE Trans. Power Syst., early access.

[14] R. Nellikkath, and S. Chatzivasileiadis, "Physics-Informed Neural Networks for AC Optimal Power Flow", Electr. Power Syst. Res., Vol. 212, pp. 108412, November 2022.

[15] F. Ferdinando, T. W. Mak, and P. V. Hentenryck. "Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods." Proceedings of the AAAI conference on artificial intelligence, Vol. 34. No. 01. 2020.

[16] W. Huang, X. Pan, M. Chen and S. H. Low, "DeepOPF-V: Solving AC-OPF Problems Efficiently," IEEE Trans. Power Syst., vol. 37, no. 1, pp. 800-803, Jan. 2022.

[17] K. Baker, "Learning Warm-Start Points For Ac Optimal Power Flow," 2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP), Pittsburgh, PA, USA, 2019, pp. 1-6.

[18] X. Pan, T. Zhao, M. Chen and S. Zhang, "DeepOPF: A Deep Neural Network Approach for Security-Constrained DC Optimal Power Flow," IEEE Trans. Power Syst., vol. 36, no. 3, pp. 1725-1735, May 2021.

[19] M. Li, S. Kolouri, and J. Mohammadi, "Learning to Solve Optimization Problems With Hard Linear Constraints," IEEE Access, vol. 11, pp. 59 995–60 004, 2023.

[20] E. Liang, M. Chen, and S. Low. "Low Complexity Homeomorphic Projection to Ensure Neural-Network Solution Feasibility for Optimization over (Non-) Convex Set." International Conference on Machine Learning, 2023.

[21] M. Kim and H. Kim, "Projection-aware deep neural network for dc optimal power flow without constraint violations," 2022 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), 2022, pp. 116–121.

[22] P. Donti, D. Rolnick, and J. Z. Kolter, "DC3: A learning method for optimization with hard constraints," International Conference on Learning Representations, 2021.

[23] K. Hongseok, "Self-supervised Equality Embedded Deep Lagrange Dual for Approximate Constrained Optimization." arXiv preprint arXiv:2306.06674 (2023).

[24] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," International Conference on Machine Learning. PMLR, 2017, pp. 136–145.

[25] L. Guo et al., "Data-Driven Power Flow Calculation Method: A Lifting Dimension Linear Regression Approach," IEEE Trans. Power Syst., vol. 37, no. 3, pp. 1798-1808, May 2022.

[26] L. Guo et al., "Model-free optimal volt-VAR control of wind farm based on data-driven lift-dimension linear power flow," CSEE J. Power Energy Syst, early access.

[27] C. Crozier and K. Baker, "Data-driven Probabilistic Constraint Elimination for Accelerated Optimal Power Flow," 2022 IEEE Power & Energy Society General Meeting (PESGM), Denver, CO, USA, 2022, pp. 1-5.

[28] E. Diao, et al. "Pruning deep neural networks from a sparsity perspective." arXiv preprint arXiv:2302.05601 (2023).

[29] M. Chatzos, T. W. K. Mak and P. V. Hentenryck, "Spatial Network Decomposition for Fast and Scalable AC-OPF Learning," IEEE Trans. Power Syst., vol. 37, no. 4, pp. 2601-2612, July 2022.

[30] Y. Li, W. Li, W. Yan, J. Yu and X. Zhao, "Probabilistic Optimal Power Flow Considering Correlations of Wind Speeds Following Different Distributions," IEEE Trans. Power Syst., vol. 29, no. 4, pp. 1847-1854, July 201.