Unsupervised Learning for Solving AC Optimal Power Flows: Design, Analysis, and Experiment

Wanjun Huang, Minghua Chen, *Fellow*, *IEEE*, and Steven H. Low, *Fellow*, *IEEE*

***Abstract*—AC optimal power flow (AC-OPF) problems need to be solved more frequently in the future for reliable and economic power system operation. Supervised-learning approaches have been developed to solve AC-OPF problems fast and accurately by learning the load-solution mapping. However, due to the non-convexity of AC-OPF problems, it is non-trivial and computationally expensive to prepare a large training dataset, and multiple load-solution mappings may exist to impair the learning even if the dataset is available. We develop an *unsupervised* learning approach (DeepOPF-NGT) for solving AC-OPF problems without the need of training dataset with ground truths to operate. It uses a properly designed loss function to guide neural networks to directly learn a valid load-solution mapping. To tackle the unbalanced gradient pathologies known to deteriorate the *unsupervised* learning performance, we develop an adaptive learning rate algorithm to dynamically balance the gradient contributions from different loss terms during training. We also derive the first condition for *unsupervised* learning to learn a valid load-solution mapping and avoid the multiple mapping issue in supervised learning. Results of IEEE 39/118/300-bus systems verify that DeepOPF-NGT can achieve comparable optimality, feasibility, and speedup performance against the state-of-the-art supervised-learning approaches, and a few ground truths can improve DeepOPF-NGT.**

***Index Terms*—AC optimal power flow, deep neural network, unsupervised learning, adaptive learning rate.**

I. Introduction

The essential AC optimal power flow (AC-OPF) problem has been widely studied to minimize operational cost by optimizing power generation subject to the physical and operational constraints. A study by FERC shows that an efficient AC-OPF solution can potentially save tens of billions of dollars every year [1]. With the increasing penetration of intermittent renewables such as wind and solar power, the AC-OPF problem will need to be solved more frequently to maintain the stable and economic operation of power systems. As a result, it is of great interest to solve AC-OPF problems with high efficiency.

There has been extensive research on efficient solution methods for AC-OPF problems. Over the last few decades, various mathematical optimization techniques have been proposed to simplify the system model (e.g., linearization), reduce the problem size (e.g., distributed algorithms), or accelerate the convergence (e.g., warm-start point) [2]. Since these methods need to solve optimization problems repeatedly for the same system with varying loads, their computational speedup is limited. Recently, machine learning methods have attracted growing attention for their promising results [3], e.g., a computational speedup up to 2$\sim$3 orders of magnitude faster than conventional solvers [**?**], [4].

Most machine learning schemes apply supervised-learning techniques to learn the load-solution mapping based on a given dataset generated by physics-based solvers such as Matpower Interior Point Solver (MIPS). Existing approaches roughly fall into two main categories: hybrid and stand-alone approaches. The hybrid approach aims to accelerate the convergence of conventional solvers by predicting hot-start points [5] or active [6]/inactive constraints [7]. Since it still needs to solve the AC-OPF problem iteratively, its computational speedup is limited. The stand-alone approach predicts decision variables and then reconstructs the remaining variables using a power flow solver [**?**], [8]–[12]. Without solving the optimization problem, it gains higher computational speedup than the hybrid approach. The study in [13] combines deep neural networks (DNNs) and Lagrangian duality to predict all variables directly and achieves an even larger computational speedup. But its solution may not satisfy Kirchhoff's circuit laws. To improve both computational speedup and feasibility, a DNN-based voltage-constrained approach, DeepOPF-V, predicts all bus voltages and then reconstructs the remaining variables via simple matrix operation [4].

Despite the promising results above, the supervised-learning approach has two main limitations. First, it is non-trivial and computationally expensive to prepare a large dataset to train DNNs to achieve considerable accuracy, as the AC-OPF problems are non-convex and hard to solve, especially for large-scale instances [14]. Second, even if one can afford to prepare a large dataset, DNNs may still fail to learn well as the training data may come from different load-(sub-)optimal-solution mappings (see Section II-B for an illustrating example). The reason is that due to the non-convexity of AC-OPF problems, existing solvers may provide one of the globally/locally optimal solutions, and different initial points may even lead to different solutions [15]. It is difficult to prepare a dataset containing data from only one mapping. The study in [16] prepares ground truths from only one load-solution mapping by solving a bi-level optimization problem, but finding the exact solution of the problem is computationally prohibitive due to the NP-hardness.

To tackle both issues above, we develop an *unsupervised* learning approach DeepOPF-NGT to solve AC-OPF problems efficiently without ground truths. Although there have been reinforcement learning methods [17], [18], they rely on previous system states and focus on total costs over a long period of time. Our approach searches the load-solution mapping without the guidance of ground truths and avoids learning the average of multiple mappings (i.e., multiple global or local optimal solutions for each input) that may lead to bad performance. In particular, we prove that DeepOPF-NGT can learn a valid

mapping (that gives a unique global or local optimal solution for each input) under certain conditions. The main contributions of this paper are three-fold:

- An *unsupervised* learning approach, DeepOPF-NGT, is proposed to solve AC-OPF problems. It leverages a properly designed loss function consisting of the objective and the penalties for constraint violations to guide DNNs to directly learn a valid load-solution mapping.
- An adaptive learning rate algorithm is designed to tackle the unbalanced gradient pathologies known to deteriorate the performance of *unsupervised* learning [19] by dynamically balancing the gradient contributions from different terms in the loss function. We provide theoretical insights into learning a valid mapping for optimization problems with multiple mappings by the *unsupervised* learning method. To the best of our knowledge, this gives the first justification of applying *unsupervised* learning to learn a valid mapping and avoid the multiple mapping issue in supervised learning (see Section II-B for an illustrating example). Furthermore, we consider the situation with a few ground truths and leverage them to improve DeepOPF-NGT by designing a semi-supervised learning approach.
- Simulations of modified IEEE 39/118/300 bus systems verify that: DeepOPF-NGT achieves comparable performances as several existing supervised-learning approaches when there is only one global/local optimal solution for each load input in their datasets, but achieves better performances than them when there are multiple global/local optimal solutions for each load input in their datasets; A few ground truths can improve the training efficiency of DeepOPF-NGT; The proposed adaptive learning rate algorithm can alleviate the unbalanced gradient pathologies.

DeepOPF-NGT is a general *unsupervised* learning framework that can be extended to other *unsupervised* regression problems. It considerably extends our preliminary work in [**?**][1] with three important improvements: (i) It improves the training efficiency significantly by the proposed adaptive learning rate algorithm, which is critical for large-scale systems; (ii) It explores the multiple mapping features of the AC-OPF problem and provides theoretical insights into learning a valid mapping; (iii) It extends DeepOPF-NGT to a semi-supervised-learning approach.

Noticeably, DeepOPF-NGT differs from a recent supervised/unsupervised scheme DC3 [20] in several aspects. First, DeepOPF-NGT has a much larger speedup than DC3. The DC3 learns the "load-generation" mapping and reconstructs the remaining variables by solving power flow equations, while DeepOPF-NGT learns the "load-voltage" mapping and directly reconstructs the remaining variables via simple matrix operation. Second, the training of DeepOPF-NGT is simpler than DC3. In DC3, the gradient of the loss function is implicit, and the inverse of the Jacobian matrix needs to be computed, whereas in DeepOPF-NGT, the gradient of the loss function is explicit, and we only need to compute the Jacobian matrix. Third, DC3 uses fixed coefficients for different terms in the loss function, while DeepOPF-NGT devises an adaptive learning rate algorithm to tackle the gradient pathologies [19]. The DC3 only shows experimental results in a small system, i.e., a 57-bus system, while DeepOPF-NGT has been validated to have good performance in the IEEE 118-bus and 300-bus systems. Fourth, we prove for the first time in the literature that the *unsupervised* learning method can guarantee to learn a valid mapping under certain conditions. Fifth, we extend DeepOPF-NGT to a semi-supervised learning approach.

We also notice two recent studies on *unsupervised* learning methods in [21], [22]. In the strict sense, the method in [21] is not *unsupervised* learning, because it still needs a large training dataset with feasible solutions. Additionally, it evaluates the feasibility of the solution approximately by DNNs, which may not be accurate. Similar to [**?**], the study in [22] uses a loss function consisting of the objective and constraint violation penalties to train the graph neural networks. Yet it does not consider the unbalanced gradient pathologies that are known to deteriorate the performance of *unsupervised* learning [19].

The structure of the remaining part is as follows. Section II formulates the AC-OPF problem and explains the multiple load-solution mapping issue. Section III illustrates the framework of DeepOPF-NGT. Section IV analyzes the performance of DeepOPF-NGT and extends it to a semi-supervised learning approach. The effectiveness of DeepOPF-NGT is verified by case studies in Section V. Section VI concludes this paper and discusses future directions.

II. Problem Formulation

A. AC Optimal Power Flow Model

Using the bus injection model, the standard AC-OPF model can be formulated as follows:

$$\min \sum\nolimits_{i\in\mathcal{N}_G} C_i(P_{gi}), \tag{1a}$$

$$\text{s.t. } P_{ij} = g_{ij}V_i^2 - V_iV_j(b_{ij}\sin\theta_{ij} + g_{ij}\cos\theta_{ij}), \forall (i,j)\in\mathcal{E}, \tag{1b}$$

$$Q_{ij} = -b_{ij}V_i^2 - V_iV_j(g_{ij}\sin\theta_{ij} - b_{ij}\cos\theta_{ij}), \forall (i,j)\in\mathcal{E}, \tag{1c}$$

$$P_i = \sum\nolimits_{(i,j)\in\mathcal{E}} P_{ij}, \quad \forall i\in\mathcal{N}, \tag{1d}$$

$$Q_i = \sum\nolimits_{(i,j)\in\mathcal{E}} Q_{ij}, \quad \forall i\in\mathcal{N}, \tag{1e}$$

$$P_i = P_{gi} - P_{di}, \quad \forall i\in\mathcal{N}, \tag{1f}$$

$$Q_i = Q_{gi} - Q_{di}, \quad \forall i\in\mathcal{N}, \tag{1g}$$

$$\underline{P}_{gi} \le P_{gi} \le \overline{P}_{gi}, \quad \forall i\in\mathcal{N}, \tag{1h}$$

$$\underline{Q}_{gi} \le Q_{gi} \le \overline{Q}_{gi}, \quad \forall i\in\mathcal{N} \tag{1i}$$

$$\underline{V}_i \le V_i \le \overline{V}_i, \quad \forall i\in\mathcal{N} \tag{1j}$$

$$\underline{\theta}_{ij} \le \theta_{ij} \le \overline{\theta}_{ij}, \quad \forall (i,j)\in\mathcal{E}, \tag{1k}$$

$$P_{ij}^2 + Q_{ij}^2 \le \overline{S}_{ij}^2, \quad \forall (i,j)\in\mathcal{E}, \tag{1l}$$

where $\mathcal{N}$ and $\mathcal{E}$ denote the sets of buses and branches, respectively; $\mathcal{N}_G$ is the set of generation buses; the branch from bus i to bus j is represented as $(i,j)\in\mathcal{E}$; g_{ij} and b_{ij} are conductance

[1] This workshop does not publish proceedings, and submissions are non-archival. Submission to this workshop does not preclude future publication.

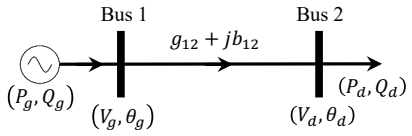


Fig. 1. An illustrating two-bus system: (P_g, Q_g) are active and reactive power generations, (P_d, Q_d) are active and reactive loads, and (V_g, θ_g) and (V_d, θ_d) are voltages of generation and load buses, respectively.

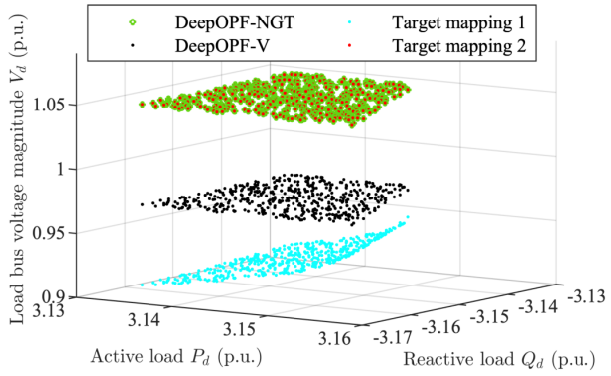


Fig. 2. Load-solution mappings in an illustrating two-bus system.

and susceptance of the branch (i, j), respectively; (P_i, Q_i) are the net active and reactive power injections, (P_{gi}, Q_{gi}) are the active and reactive power generations, (P_{di}, Q_{di}) are the active and reactive loads, (V_i, θ_i) are the voltage magnitude and angle, at bus i; $\theta_{ij} = \theta_i - \theta_j$ denotes the branch angle, and (P_{ij}, Q_{ij}) are the active and reactive branch power flows, at branch (i, j). The upper and lower bounds of a variable x are represented by $\overline{x}$ and $\underline{x}$, respectively. The AC-OPF problem aims to minimize the generation cost in (1a) with all physical constraints in (1b)-(1l) satisfied. The branch power flows are given by (1b)-(1c) and restricted by (1l); the Kirchhoff's circuit laws are ensured by (1d)-(1e); net active and reactive power injections are given by (1f)-(1g), respectively; active and reactive power generation limits are enforced by (1h)-(1i); voltage magnitudes and angles are restricted by (1j) and (1k), respectively.

B. Multiple Mappings for the AC-OPF Problem

For a given load input, the optimal solution of the AC-OPF problem is obtained by solving the formulation in (1). There may be multiple local or global optimums due to the non-convexity of the problem. Existing OPF solvers such as MIPS may provide different local or global optimal solutions when starting from different initial points. Hence, the dataset generated by the solver may contain mixed samples from different load-solution mappings. We present an example of a two-bus system with one generation bus and one load bus [23] (see Fig. 1). We applied MIPS to generate 1000 training samples and 500 test samples with different load inputs. The method in [23] is used to find multiple global (local) solutions. For each load input, we generate 2000 different initial points within the bounds of each variable using uniform sampling. Then, we solve the AC-OPF problem in (1) starting from each initial point. The branch resistance is set to a small value such that the costs of different solutions are almost the same for each load input.

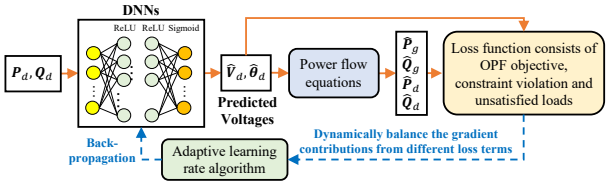


Fig. 3. Schematic of the proposed DeepOPF-NGT.

The test samples are shown in Fig. 2. The mapping between loads and voltage magnitudes is shown as an example. In Fig. 2, there are two solutions with almost the same costs for each load input, which forms two valid load-solution mappings, i.e., target mappings 1 and 2 marked by blue and red dots in Fig. 2, respectively. When using the supervised-learning approach, the samples generated from the solver may come from both target mappings, resulting in learning neither of the two mappings. However, this is in principle not a concern for the *unsupervised* learning as it does not require ground truth to operate.

We compared the supervised-learning approach DeepOPF-V in [4] and DeepOPF-NGT in learning the load-solution mapping using the dataset introduced above. Due to the space limitation, we only show the predicted load bus voltage magnitude in Fig. 2. The mappings learned by DeepOPF-NGT and DeepOPF-V are marked by green circles and black dots, respectively. The mapping learned by DeepOPF-NGT is almost the same as target mapping 2 (which has slightly smaller costs than target mapping 1), with all constraints and 99.85% active loads satisfied, while that learned by DeepOPF-V is located between the two target mappings, with all constraints but only 98.8% active loads satisfied. This example suggests that the *unsupervised* learning approach outperforms the *supervised* learning approach in the regression task with multiple load-solution correspondences for each load input.

III. DeepOPF-NGT: An Unsupervised-Learning Approach without Ground Truth

A. Schematic of DeepOPF-NGT

The schematic of the proposed DeepOPF-NGT is shown in Fig. 3. The DNN-based model is composed of fully connected layers, with a rectified linear unit (ReLU) activation function on each hidden layer and a sigmoid activation function on the output layer. It aims to learn the mapping between load configurations $(\boldsymbol{P_d}, \boldsymbol{Q_d})$ and bus voltages $(\boldsymbol{\theta}, \boldsymbol{V})$, where $\boldsymbol{P_d}$ and $\boldsymbol{Q_d}$ are vectors of active and reactive loads, respectively, and $\boldsymbol{\theta}$ and $\boldsymbol{V}$ are vectors of voltage angles and magnitudes, respectively. Using the well-trained DNNs, the predicted voltage magnitudes $\hat{\boldsymbol{V}}$ and voltage angles $\hat{\boldsymbol{\theta}}$ can be obtained instantly with $(\boldsymbol{P_d}, \boldsymbol{Q_d})$ as the input. Next, based on $(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{V}})$ and $(\boldsymbol{P_d}, \boldsymbol{Q_d})$, we can easily compute the left-hand side of the equations of net power injections in (1f)-(1g). Then, the remaining solution variables $(\hat{\boldsymbol{P}}_{\boldsymbol{g}}, \hat{\boldsymbol{Q}}_{\boldsymbol{g}})$ and the auxiliary variables $(\hat{\boldsymbol{P}}_{\boldsymbol{d}}, \hat{\boldsymbol{Q}}_{\boldsymbol{d}})$ are directly calculated by (1f)-(1g) using the obtained left-hand side values without solving non-linear power flow equations. Specifically, for each bus i: 1) if there are only generators or loads, its predicted active and reactive generations (i.e., $\hat{P}_{gi}$ and $\hat{Q}_{gi}$) or predicted active and reactive

loads (i.e., $\hat{P}_{di}$ and $\hat{Q}_{di}$) are obtained directly; 2) if there are both generators and loads, $\hat{P}_{di}$ and $\hat{Q}_{di}$ are set to the given loads P_{di} and Q_{di}, respectively, and then $\hat{P}_{gi}$ and $\hat{Q}_{gi}$ are directly calculated from (1f)-(1g). The objective function is calculated by (1a) after obtaining $\hat{\boldsymbol{P}}_{\boldsymbol{g}}$.

To guide the *unsupervised* training of DNNs without ground truth, the loss function is designed as

$$\mathcal{L} = k_{obj}\mathcal{L}_{obj} + \mathcal{L}_{cons} + k_d\mathcal{L}_d, \tag{2}$$

where k_{obj} and k_d are positive constants, $\mathcal{L}_{obj}$ is the objective in (1a), $\mathcal{L}_{cons}$ is designed to find feasible solutions satisfying the constraints in (1h)-(1k), and $\mathcal{L}_d$ is designed to satisfy demanded loads. Kirchhoff's circuit laws in (1d)-(1e) are satisfied automatically since the net power injections can always be calculated with the predicted bus voltages. We apply $\mathcal{L}$ to guide train DNN training. Specifically, $\mathcal{L}_{cons}$ is the penalty for constraint violations during training as below:

$$\mathcal{L}_{cons} = k_g\mathcal{L}_g + k_{S_l}\mathcal{L}_{S_l} + k_{\theta_l}\mathcal{L}_{\theta_l}, \tag{3}$$

where k_g, k_{S_l} and k_{θ_l} are positive constants; $\mathcal{L}_g$, $\mathcal{L}_{S_l}$ and $\mathcal{L}_{\theta_l}$ are penalties for the violations of generation, branch flow and branch angle constraints during training, respectively, i.e.,

$$\begin{aligned}\mathcal{L}_g = \sum\nolimits_{i\in\mathcal{N}_G}[&\max((\hat{P}_{gi}-\overline{P}_{gi})^2,0)+\max((\underline{P}_{gi}-\hat{P}_{gi})^2,0)\\&+\max((\hat{Q}_{gi}-\overline{Q}_{gi})^2,0)+\max((\underline{Q}_{gi}-\hat{Q}_{gi})^2,0)],\end{aligned} \tag{4}$$

$$\mathcal{L}_{S_l} = \sum\nolimits_{(i,j)\in\mathcal{E}}[\max((\hat{S}_{ij}-\overline{S}_{ij})^2,0)], \tag{5}$$

$$\mathcal{L}_{\theta_l} = \sum\nolimits_{(i,j)\in\mathcal{E}}[\max((\hat{\theta}_{ij}-\overline{\theta}_{ij})^2,0)+\max((\underline{\theta}_{ij}-\hat{\theta}_{ij})^2,0)]. \tag{6}$$

Here, $\hat{S}_{ij}=\sqrt{\hat{P}_{ij}^2+\hat{Q}_{ij}^2}$, and $(\hat{P}_{ij},\hat{Q}_{ij})$ are branch active and reactive power flows derived from predicted voltages $(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{V}})$. The term $\mathcal{L}_d$ penalizes the deviations between the actual loads $(\boldsymbol{P_d},\boldsymbol{Q_d})$ and the predicted loads $(\hat{\boldsymbol{P}}_{\boldsymbol{d}},\hat{\boldsymbol{Q}}_{\boldsymbol{d}})$ as

$$\mathcal{L}_d = \sum\nolimits_{i\in\mathcal{N}_L}\left[(\hat{P}_{di}-P_{di})^2+(\hat{Q}_{di}-Q_{di})^2\right], \tag{7}$$

where $\mathcal{N}_L$ is the set of load buses. To dynamically balance the gradient contributions from different loss terms in (2) in DNN training, we design an adaptive learning rate algorithm in Section. III-B. The post-processing method in [4] is used to improve the feasibility of the predicted solution.

As discussed in [4], there could be mismatches between the demanded loads $(\boldsymbol{P_d},\boldsymbol{Q_d})$ and the satisfied loads $(\hat{\boldsymbol{P}}_{\boldsymbol{d}},\hat{\boldsymbol{Q}}_{\boldsymbol{d}})$ due to prediction errors. Unsatisfied loads are also inevitable in conventional methods. Considering power losses in transmission lines, inexact system parameters, etc., a small load-generation imbalance ratio (around 1%) is acceptable [24].

B. Neural Network Training of DeepOPF-NGT

The DNN model of DeepOPF-NGT is trained by minimizing the loss function $\mathcal{L}$ in (2). A simple and viable approach is to apply the gradient descent algorithm. Let us denote the DNN parameters as $\boldsymbol{\phi}$ and the mapping from the input $\boldsymbol{x}=(\boldsymbol{P_d},\boldsymbol{Q_d})^T$ to the output $\boldsymbol{y}=(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{V}})$ as $\boldsymbol{y}=\xi(\boldsymbol{x},\boldsymbol{\phi})$. Then, we can obtain $\hat{\boldsymbol{P}}_{\boldsymbol{g}}(\xi(\boldsymbol{x},\boldsymbol{\phi}))$, $\hat{\boldsymbol{Q}}_{\boldsymbol{g}}(\xi(\boldsymbol{x},\boldsymbol{\phi}))$, $\hat{\boldsymbol{P}}_{\boldsymbol{d}}(\xi(\boldsymbol{x},\boldsymbol{\phi}))$, $\hat{\boldsymbol{Q}}_{\boldsymbol{d}}(\xi(\boldsymbol{x},\boldsymbol{\phi}))$, $\hat{\boldsymbol{S}}_{\boldsymbol{l}}(\xi(\boldsymbol{x},\boldsymbol{\phi}))$ and $\hat{\boldsymbol{\theta}}_{\boldsymbol{l}}(\xi(\boldsymbol{x},\boldsymbol{\phi}))$, where $\hat{\boldsymbol{S}}_{\boldsymbol{l}}=(\hat{S}_{ij},\forall(i,j)\in\mathcal{E})$ and $\hat{\boldsymbol{\theta}}_{\boldsymbol{l}}=(\hat{\theta}_{ij},\forall(i,j)\in\mathcal{E})$. The DNN parameters $\boldsymbol{\phi}$ are updated according to $\boldsymbol{\phi}_{t+1}=\boldsymbol{\phi}_t-\eta_t\nabla_{\boldsymbol{\phi}_t}\mathcal{L}$, where η_t is a positive step size at the t-th epoch of training, and the gradient $\nabla_{\boldsymbol{\phi}}\mathcal{L}$ is obtained by using chain rule as below:

$$\begin{aligned}&\nabla_{\boldsymbol{\phi}}\mathcal{L}=\nabla_{\boldsymbol{y}}\mathcal{L}\cdot\nabla_{\boldsymbol{\phi}}\boldsymbol{y}\\&=(k_{obj}\nabla_{\boldsymbol{y}}\mathcal{L}_{obj}+\nabla_{\boldsymbol{y}}\mathcal{L}_{cons}+k_d\nabla_{\boldsymbol{y}}\mathcal{L}_d)\cdot\nabla_{\boldsymbol{\phi}}\boldsymbol{y}\\&=[(k_{obj}\nabla_{\hat{\boldsymbol{P}}_{\boldsymbol{g}}}\mathcal{L}_{obj}+k_g\nabla_{\hat{\boldsymbol{P}}_{\boldsymbol{g}}}\mathcal{L}_g)\cdot\nabla_{\boldsymbol{y}}\hat{\boldsymbol{P}}_{\boldsymbol{g}}(\boldsymbol{y})\\&+k_g\nabla_{\hat{\boldsymbol{Q}}_{\boldsymbol{g}}}\mathcal{L}_g\cdot\nabla_{\boldsymbol{y}}\hat{\boldsymbol{Q}}_{\boldsymbol{g}}(\boldsymbol{y})+k_{S_l}\nabla_{\hat{\boldsymbol{S}}_{\boldsymbol{l}}}\mathcal{L}_{S_l}\cdot\nabla_{\boldsymbol{y}}\hat{\boldsymbol{S}}_{\boldsymbol{l}}(\boldsymbol{y})\\&+k_{\theta_l}\nabla_{\boldsymbol{\theta_l}}\mathcal{L}_{\theta_l}\cdot\nabla_{\boldsymbol{y}}\hat{\boldsymbol{\theta}}_{\boldsymbol{l}}(\boldsymbol{y})+k_d\nabla_{\hat{\boldsymbol{P}}_{\boldsymbol{d}}}\mathcal{L}_d\cdot\nabla_{\boldsymbol{y}}\hat{\boldsymbol{P}}_{\boldsymbol{d}}(\boldsymbol{y})\\&+k_d\nabla_{\hat{\boldsymbol{Q}}_{\boldsymbol{d}}}\mathcal{L}_d\cdot\nabla_{\boldsymbol{y}}\hat{\boldsymbol{Q}}_{\boldsymbol{d}}(\boldsymbol{y})]\cdot\nabla_{\boldsymbol{\phi}}\boldsymbol{y}.\end{aligned} \tag{8}$$

All terms in $\nabla_{\boldsymbol{y}}\mathcal{L}$ can be derived from (1) and (4)-(7). For faster convergence, we employ the mini-batch stochastic gradient descent algorithm [25].

The key challenge of training DeepOPF-NGT lies in gradient pathologies mainly caused by the gradient imbalance between different loss terms in (2), a common issue when applying gradient decent method [19]. The dominant loss term may bias the DNN training towards neglecting the contribution of others. To address this issue, we develop an adaptive learning rate algorithm based on the learning rate annealing algorithm developed in [19] for solving non-linear partial differential equations. Different from the algorithm in [19], we directly balance different loss terms and add upper bounds (fine-tuned manually) to their coefficients to prevent gradient explosion. For each training epoch $t>1$, given a fixed coefficient k_{obj}, the coefficients k_g^t, $k_{S_l}^t$, $k_{\theta_l}^t$ and k_d^t are updated as follows:

$$k_g^t = \min(k_{obj}\mathcal{L}_{obj}/\mathcal{L}_g,\overline{k}_g), \tag{9a}$$

$$k_{S_l}^t = \min(k_{obj}\mathcal{L}_{obj}/\mathcal{L}_{S_l},\overline{k}_{S_l}), \tag{9b}$$

$$k_{\theta_l}^t = \min(k_{obj}\mathcal{L}_{obj}/\mathcal{L}_{\theta_l},\overline{k}_{\theta_l}), \tag{9c}$$

$$k_d^t = \min(k_{obj}\mathcal{L}_{obj}/\mathcal{L}_d,\overline{k}_d), \tag{9d}$$

where $\overline{k}_g$, $\overline{k}_{S_l}$, $\overline{k}_{\theta_l}$, and $\overline{k}_d$ are the upper bounds of k_g^t, $k_{S_l}^t$, $k_{\theta_l}^t$ and k_d^t, respectively. Then, we developed Algorithm 1 for the training of DeepOPF-NGT.

Algorithm 1: Training process of DeepOPF-NGT

Input : DNN model with initial parameters $\boldsymbol{\phi}_0$, other initial parameters η_0, k_{obj}, k_g^0, $k_{S_l}^0$, $k_{\theta_l}^0$, and k_d^0, maximum training epoch T, training dataset $\mathcal{D}=\{\boldsymbol{x}_1,...,\boldsymbol{x}_n\}$.

Output : Trained DNN model with parameters $\boldsymbol{\phi}$.

```
for t = 1 to T do
    Shuffle the training dataset D.
    for each batch B ⊂ D do
        for i = 1 to m do
            ŷ_i ← ξ̂(x_i, φ_{t−1})
        end
        Compute L_obj, L_g, L_{S_l}, L_{θ_l} and L_d.
        Update k_g^t, k_{S_l}^t, k_{θ_l}^t, k_d^t by (9) and calculate L.
        φ_t ← φ_{t−1} − η_t ∇_{φ_{t−1}} L
    end
    t ← t + 1
end
```

C. Discussion

The optimal parameters of DNNs are searched by using the mini-batch gradient descent algorithm. On one hand, due to the non-convexity of the loss function $\mathcal{L}$, the obtained parameters of DNNs may be sub-optimal. Hence, the predicted AC-OPF solutions may be sub-optimal. However, existing solvers can only provide sub-optimal solutions as well. Up till now, how to find globally optimal solutions for AC-OPF problems under general settings is still an open problem. On the other hand, compared with existing supervised-learning methods, DeepOPF-NGT does not need ground truths and has the potential to find better solutions than conventional solvers.

IV. Performance Analysis and Extension of DeepOPF-NGT

A. Learning a Valid Mapping

In this subsection, we explore sufficient conditions for learning a valid mapping (see Definiton 3) by the proposed *unsupervised* learning approach DeepOPF-NGT. We start by formalizing the notions. We consider a standard AC-OPF problem and assume there are M continuous target load-solution mappings. Each target mapping is denoted as $\xi_i : \boldsymbol{x} \to \boldsymbol{y}, i = 1, 2, ...M$, and the mapping from $\boldsymbol{y}$ to the loss function $\mathcal{L}$ is represented by $\psi : \boldsymbol{y} \to \mathcal{L}$, where $\boldsymbol{x} \in \mathbb{R}^{2N_L}$ and $\boldsymbol{y} \in \mathbb{R}^{2N}$ are the input (loads) and the output (ground truths of voltage magnitudes and angles), respectively. N_L and N are cardinalities of $\mathcal{N}_L$ and $\mathcal{N}$, respectively. The feasible sets of $\boldsymbol{x}$, $\boldsymbol{y}$ are denoted as $\mathcal{X}$ and $\mathcal{Y}$, respectively. The set of all target mappings is denoted as $\boldsymbol{\xi}$. The mapping learned by DNNs is denoted as ξ_{nn}. In the compact set $\mathcal{X}$, ξ_i is Lipschitz continuous except for a set of Lebesgue measure zero [**?**], and ξ_{nn} is Lipschitz continuous over $\mathcal{X}$ [9].

Regarding ψ, we make the following assumption:

- A1: $\psi(\boldsymbol{y})$ is a strongly monotone operator, i.e., there exists a constant $L_\psi > 0$ such that $\big(\psi(\boldsymbol{y}_i) - \psi(\boldsymbol{y}_j)\big)^T (\boldsymbol{y}_i - \boldsymbol{y}_j) \geq L_\psi \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2$ for any $\boldsymbol{y}_i, \boldsymbol{y}_j \in \mathcal{Y}$ [26].

Assumption A1 requires ψ to be smooth for DNNs to learn the target mapping well. By A1, we have Lemma 1 as follows, which limits how fast ψ changes and gives the lower bound.

Lemma 1. *Suppose A1 holds. There exists a constant $L_\psi > 0$ such that for any $\boldsymbol{y}_i, \boldsymbol{y}_j \in \mathcal{Y}$, one has*

$$\|\psi(\boldsymbol{y}_i) - \psi(\boldsymbol{y}_j)\|_2 \geq L_\psi \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2 \tag{10}$$

Proof: See Appendix A. ∎

Before proceeding, we have the following definitions for our analysis. First, we define whether two target mappings ξ_i and ξ_j can be distinguished according to their distances.

Definition 1. *For a given $\varepsilon > 0$ and a compact set $\mathcal{X}$, we say two mappings ξ_i and ξ_j are ε-similar to each other in $\mathcal{X}$ if $\|\xi_i(\boldsymbol{x}) - \xi_j(\boldsymbol{x})\|_2 \leq \varepsilon$ for all $\boldsymbol{x} \in \mathcal{X}$.*

Note that "ε-similar" means "not ε-distinguishable". In AC-OPF problems, we observed that in some subsets of $\mathcal{X}$, there exists $\varepsilon > 0$ such that ξ_i and ξ_j are ε-distinguishable, while in other subsets of $\mathcal{X}$, we can not find $\varepsilon > 0$ such that ξ_i and ξ_j can be distinguished. Thus, we further define the ε-similar set.

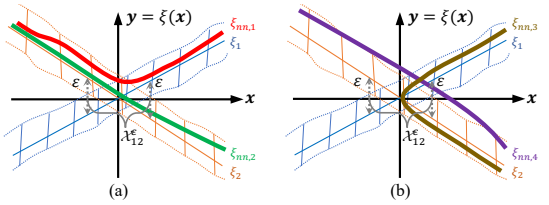


Fig. 4. An example of a single-input single-output mapping: ξ_1 and ξ_2 are target mappings, while $\xi_{nn,1}$ and $\xi_{nn,2}$ are ε-similar valid mappings.

Definition 2. *For a given $\varepsilon > 0$ and two mappings $\xi_i, \xi_j \in \boldsymbol{\xi}$ in a compact set $\mathcal{X}$, the ε-similar set $\mathcal{X}_{ij}^{\varepsilon} \subseteq \mathcal{X}$ is the set where ξ_i and ξ_j are ε-similar to each other.*

Based on the above, we define the valid mapping as follows.

Definition 3. *For a given $\delta > 0$ and a compact set $\mathcal{X}$, ξ_{nn} is a valid mapping, if there exists one and only one target mapping $\xi_i \in \boldsymbol{\xi}$ such that $\|\psi(\xi_{nn}(\boldsymbol{x})) - \psi(\xi_i(\boldsymbol{x}))\|_2 \leq \delta$ for all $\boldsymbol{x} \in \mathcal{X}$.*

By Lemma 1 and Definition 3, in a valid mapping ξ_{nn}, for each $\boldsymbol{x} \in \mathcal{X}$, there exists one and only one $\xi_i \in \boldsymbol{\xi}$ such that ξ_{nn} and ξ_i are ε-similar to each other, where $\varepsilon = \delta / L_\psi$. Fig. 4 presents an example of valid mapping, where ξ_1 (marked by blue curve) and ξ_2 (marked by orange curve) are two different target mappings. In the orange dash area, $\|\psi(\xi_{nn,1}(\boldsymbol{x})) - \psi(\xi_1(\boldsymbol{x}))\|_2 \leq \delta$; In the blue dash area, $\|\psi(\xi_{nn,2}(\boldsymbol{x})) - \psi(\xi_2(\boldsymbol{x}))\|_2 \leq \delta$. By Definition 3, $\xi_{nn,1}$ (marked by green curve) and $\xi_{nn,2}$ (marked by red curve) are valid mappings, while $\xi_{nn,3}$ (marked by brown curve) and $\xi_{nn,4}$ (marked by purple curve) are not valid mappings.

Based on the above definitions, we derive sufficient conditions for DNNs to learn a valid mapping in Theorem 2. Without loss of generality, we focus on the multiple-input single-output mapping[2].

Theorem 2. *Suppose A1 holds and $\mathcal{X}$ is compact. ξ_{nn} is a valid mapping if there exists $\varepsilon > 0$ and $\delta > 0$ such that C1–C3 are satisfied:*

C1: $\forall \boldsymbol{x} \in \mathcal{X} \backslash \mathcal{X}_{ij}^{\varepsilon}$, $\forall \xi_i, \xi_j \in \boldsymbol{\xi}$, $\|\xi_i(\boldsymbol{x}) - \xi_j(\boldsymbol{x})\|_2 \geq \varepsilon$.

C2: $\forall \boldsymbol{x} \in \mathcal{X}$, $\exists \xi_i \in \boldsymbol{\xi}$, $\|\psi(\xi_{nn}(\boldsymbol{x})) - \psi(\xi_i(\boldsymbol{x}))\|_2 \leq \delta$.

C3: $\delta < \frac{\varepsilon L_\psi}{2}$.

Proof: See Appendix B. ∎

In Theorem 2, for $\boldsymbol{x} \in \mathcal{X} \backslash \mathcal{X}_{ij}^{\varepsilon}$, C1 ensures that the target mappings can be distinguished from each other by limiting the deviation between the zero-order derivatives of any two target mappings $\xi_i, \xi_j \in \boldsymbol{\xi}$; for $\boldsymbol{x} \in \mathcal{X}_{ij}^{\varepsilon}$, ξ_i and ξ_j are too close to each other to be distinguished. Condition C2 restricts the deviation between the loss functions of ξ_i and ξ_{nn}, which enforces the optimality of DNNs. For ξ_i, $\psi(\xi_i(\boldsymbol{x}))$ is the objective, whereas for ξ_{nn}, $\psi(\xi_{nn}(\boldsymbol{x}))$ consists of the objective and penalties for constraint violations and unsatisfied loads. Thus, DNNs learn the target mapping ξ_i when $\delta = 0$. Condition C3 specifies the relationship between the distance between target mappings and the DNN learning performance. It implies that it is easier for DNN to learn a valid mapping when target mappings are sufficiently different from each other.

[2] The multiple-input multiple-output mapping can be obtained by combing all multiple-input single-output mappings.

To our best knowledge, this is the first theoretical justification of learning a valid mapping for non-convex optimization problems by *unsupervised* learning. It demonstrates the potential of the proposed *unsupervised* learning approach in addressing the fundamental multiple mapping issues in the supervised-learning approach. In application, ε is determined by target mappings. As for δ, as long as DNNs achieve good learning performance (i.e., small loss function), δ will be sufficiently small for C2–C3 to be satisfied.

B. Extension to Semi-Supervised Learning Approach

Now we consider the situation where there are a few samples with ground truths. Since the ground truth contains the information of the test system, it can be leveraged to pre-train the DNN model of DeepOPF-NGT instead of learning from scratch. In the pre-training process, the supervised-learning approach is applied with the loss function as below:

$$\mathcal{L}' = k_v\mathcal{L}_v + k_g\mathcal{L}_g + k_{S_l}\mathcal{L}_{S_l} + k_{\theta_l}\mathcal{L}_{\theta_l} + k_d\mathcal{L}_d, \tag{11}$$

where k_v is a positive constant, and $\mathcal{L}_v$ is the prediction error as below:

$$\mathcal{L}_v = \sum\nolimits_{i\in\mathcal{N}} \left(||\hat{V}_i - V_i||_2^2 + ||\hat{\theta}_i - \theta_i||_2^2 \right). \tag{12}$$

In this way, DeepOPF-NGT is extended to a semi-supervised-learning approach.

The conventional pre-training method pre-trains the DNN model before the entire training process. To fully utilize the ground truths, we propose Algorithm 2 to pre-train the DNN model at each training epoch, where $\mathcal{D}$ and $\mathcal{D}'$ are datasets with and without ground truths, respectively, and $\nabla_{\boldsymbol{\phi}}\mathcal{L}'$ is derived using the chain rule. In the ideal case, the ground truths should come from the same load-solution mapping. One possible method is to use a fixed initial point when generating ground truths using a conventional OPF solver[3]. When the number of samples with ground truths is substantially smaller than that without ground truths, there may be little influence even if the ground truths are from different mappings.

C. Discussion

Distinguishing different load-solution mappings is still a challenging open problem. Thus, it is also hard to determine whether the learned mapping is a valid mapping when the conditions in Theorem 2 are not satisfied. These intriguing questions go beyond the scope of this work and provide directions for future investigation. In this study, we apply common metrics to evaluate the feasibility and optimality of DeepOPF-NGT. Although DeepOPF-NGT may not guarantee learning a valid mapping due to imperfect training in practice, the well-trained model will still achieve good optimality and feasibility performances so long as the training loss is small, as verified in Section V.

[3] It does not guarantee to generate samples from the same mapping but tries to avoid generating samples from multiple mappings. Given a fixed initial point, the deterministic solver MIPS will converge to a deterministic solution for each input [27].

Algorithm 2: Training process of extended DeepOPF-NGT

Input : DNN model with initial parameters $\boldsymbol{\phi}_0$, other initial parameters η_0, k_v, k_g^0, $k_{S_l}^0$, $k_{\theta_l}^0$, and k_d^0, maximum training epoch T, training datasets $\mathcal{D} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ and $\mathcal{D}' = \{(\boldsymbol{x}_{n+1}, \boldsymbol{y}_{n+1}), ..., (\boldsymbol{x}_{n+n'}, \boldsymbol{y}_{n+n'})\}$.

Output : Trained DNN model with parameters $\boldsymbol{\phi}$.

```
for t = 1 to T do
    Shuffle the training datasets D and D'.
    Pre-training process with ground truths:
    for each batch B' ⊂ D' do
        for i = 1 to m do
            ŷ_i ← ξ̂(x_i, φ_{t-1})
        end
        Compute L_v, L_g, L_{S_l}, L_{θ_l} and L_d.
        Update k_g^t, k_{S_l}^t, k_{θ_l}^t, k_d^t by (9) and calculate L'.
        φ'_t ← φ_{t-1} − η_t ∇_{φ_{t-1}} L'
    end
    Training process without ground truths:
    for each batch B ⊂ D do
        for i = 1 to m do
            ŷ_i ← ξ̂(x_i, φ'_t)
        end
        Compute L_obj, L_g, L_{S_l}, L_{θ_l} and L_d.
        Update k_g^t, k_s^t, k_{θ_l}^t, k_d^t by (9) and calculate L.
        φ_t ← φ'_t − η_t ∇_{φ'_t} L
    end
    t ← t + 1
end
```

V. Case Study

A. Experimental Setup

Experimental tests are conducted on the modified IEEE 39/118/300-bus systems to validate the effectiveness of DeepOPF-NGT. We compare DeepOPF-NGT with the following state-of-the-art supervised-learning approaches:

DeepOPF-V [4]: It learns the mapping between loads and voltages of all buses. For a fair comparison, we add penalties for constraint violations and load deviations to the loss function in [4], i.e., using $\mathcal{L}'$ in (11).

DeepOPF-AC [**?**]: It learns the mapping between loads and the optimal generation setpoints and then reconstructs the remaining variables by solving power balance equations.

EACOPF [11]: It trains a neural network to emulate an iterative solver. Then, it predicts the generation setpoints using the well-trained model and reconstructs the remaining variables by solving power balance equations.

The above three approaches are only compared in the modified IEEE 118-bus system due to the lengthy training time of DeepOPF-AC and EACOPF for larger systems.

The training dataset contains only a set of load scenarios for our unsupervised-learning scheme DeepOPF-NGT, but both load scenarios and the corresponding ground truths, i.e., optimal solutions of AC-OPF problems, for the bench marking supervised-learning approaches. The ground truths are

TABLE I
GENERAL PARAMETER SETTINGS FOR DNN MODELS.

System	Approach	Batch size	Learning rate	Epoch	Hidden layers
39-bus	DeepOPF-NGT	50	1e-3	12000	256-256
	DeepOPF-V	50	1e-3	12000	256-256
118-bus	DeepOPF-NGT	50	1e-3	3000	512-256
	DeepOPF-V	50	1e-3	6000	512-256
	DeepOPF-AC	32	1e-3	1000	256-128
	EACOPF	200	1e-3	3000	500
300-bus	DeepOPF-NGT	50	1e-4	10000	768-768

generated by conventional OPF solver MIPS using the interior point method. Other solvers can be used as well. To show the impact of the mixed samples from multiple mappings on the supervised-learning approaches, we generated two types of datasets. The first was generated by using the same default initial point for all load scenarios (in the IEEE 118/300-bus systems). The second was generated by using different initial points to find multiple global or local solutions for each load scenario [23] (in the IEEE 39-bus system). The dataset contains 600 training samples and 40,000 test samples for the 118/300-bus systems, and 2000 training samples and 500 test samples for the IEEE 39-bus system. There are more training samples for the second type of dataset because it contains more extreme operating conditions and binding constraints. More details of finding multiple local or global optimal solutions can be found in [23]. All load scenarios are generated by multiplying the default load by the normalized daily total load profile of Bonneville Power Administration on 02/08/2016 from 06:00 am to 12:00 pm [28]. Each experimental test was repeated three times to get the average performance.

The DNN-based models are designed on the platform Pytorch. The hyper-parameters are fine-tuned by trial and error (see Table. I). Other important hyper-parameters are set as follows: $k_v = 100$, $k_g^0 = k_{S_l}^0 = k_{\theta_l}^0 = k_d^0 = 1$. Most parameters of the comparing methods are set according to [**?**], [4], [11], [19] and fine-tuned as best as we could. The DNN-based models are all trained on a single GPU. Experimental tests are run on the quad-core (i7-3770@3.40G Hz) CPU workstation with 16GB RAM.

It is hard to visualize the high-dimensional mapping learned by DNNs. To this end, we apply the following metrics to evaluate the performance of different approaches:

1) ***Speedup Factor***: It is the average ratio of the computation time consumed by MIPS to solve the original AC-OPF problem to the computation time consumed by the DNN-based method.

2) ***Optimality Loss***: It measures the average relative deviation between the optimal objective found by MIPS and that found by the DNN-based method.

3) ***Constraint Satisfaction Ratio***: It evaluates the feasibility of the predicted solutions by the percentage of satisfied bound constraints. The constraint satisfaction ratios of active power generation, reactive power generation, bus voltage, branch power flow and branch angle are denoted by $\eta_{\boldsymbol{P_g}}$, $\eta_{\boldsymbol{Q_g}}$, $\eta_{\boldsymbol{V}}$ $\eta_{\boldsymbol{S_l}}$ and $\eta_{\boldsymbol{\theta_l}}$, respectively.

4) ***Load Satisfaction Ratio***: It is defined as the percentage of loads satisfied for the whole system. The active and reactive load satisfaction ratios are denoted by $\eta_{\boldsymbol{P_d}}$ $\eta_{\boldsymbol{Q_d}}$, respectively.

TABLE II
COMPARISON RESULTS IN THE MODIFIED IEEE 118-BUS SYSTEM.

Metric	DeepOPF-NGT	DeepOPF-V	DeepOPF-AC	EACOPF
$\eta_{opt}(\%)$	<0.1	<0.4	<0.3	<6.0
$\eta_{\boldsymbol{V}}(\%)$	-	-	99.81	**96.57**
$\eta_{\boldsymbol{P_g}}(\%)$	100.00	100.00	100.00	99.20
$\eta_{\boldsymbol{Q_g}}(\%)$	100.00	99.98	100.00	100.00
$\eta_{\boldsymbol{S_l}}(\%)$	99.43	100.00	100.00	99.46
$\eta_{\boldsymbol{\theta_l}}(\%)$	100.00	100.00	100.00	100.00
$\eta_{\boldsymbol{P_d}}(\%)$	99.30	99.91	-	-
$\eta_{\boldsymbol{Q_d}}(\%)$	99.93	99.70	-	-
η_{sp}	×1e3	×1e3	×1e2	×1e2

B. Comparison with Supervised-Learning Approaches

DeepOPF-NGT is compared with three state-of-the-art supervised-learning methods, i.e., DeepOPF-V [4], DeepOPF-AC [**?**] and EACOPF [11], in the modified IEEE 118-bus system. The load variation is up to 40%, which is larger than that (up to 10%) in [4]. Table. II shows that DeepOPF-NGT has comparable performance with the supervised-learning methods, but does not need the ground truth for DNN training.

As seen in Table. II, DeepOPF-NGT and DeepOPF-V have much greater computational speedup (up to three orders of magnitude faster than the conventional OPF solver MIPS) than DeepOPF-AC and EACOPF. That is because DeepOPF-AC and EACOP still need to solve the non-linear power balance equations, while DeepOPF-NG and DeepOPF-V only need simple matrix operations. The optimality loss of DeepOPF-NGT, DeepOPF-V and DeepOPF-AC are all less than 0.5%, while that of EACOPF is around 6.0%. Noticeably, DeepOPF-NGT has the smallest optimality loss, which may be because its loss function contains the objective of the AC-OPF problem.

Concerning the feasibility of the solution, most inequality constraints are satisfied in these four approaches. Meanwhile, DeepOPF-NGT and DeepOPF-V can ensure the satisfaction of voltage constraints, while DeepOPF-AC and EACOPF can not. The reason is that DeepOPF-NGT and DeepOPF-V obtain bus voltages directly and thus can keep them within limits, while DeepOPF-AC and EACOPF reconstruct bus voltages by solving power flow equations using predicted generation setpoints, thus the voltage constraints may be violated. Moreover, in a worst-case scenario, DeepOPF-AC and EACOPF may have no feasible power flow solutions (see results in [4]). However, the power flow equations are satisfied automatically in DeepOPF-NGT and DeepOPF-V. Thus, DeepOPF-NGT and DeepOPF-V always guarantee to obtain a solution, while DeepOPF-AC and EACOPF may have no solution. This is validated by Table. II showing that the voltage satisfaction ratios of DeepOPF-AC and EACOPF are 99.81% and 96.57%, respectively, while DeepOPF-NGT and DeepOPF-V have no voltage violation and have acceptable load satisfaction ratios over 99.70%. There is a slight violation of the branch flow constraint in DeepOPF-NGT due to the binding of one branch flow constraint.

C. Performance in Handling Multiple Mappings

To verify the effectiveness of DeepOPF-NGT in addressing the issue of multiple load-solution mappings, we compare it with DeepOPF-V in the modified IEEE 39-bus system. These two approaches are compared as they have the same input

TABLE III
COMPARISON RESULTS IN THE MODIFIED IEEE 39-BUS SYSTEM.

Metric	DeepOPF-NGT	DeepOPF-V
$\eta_{opt}(\%)$	<5	<1
$\eta_{\boldsymbol{V}}(\%)$	-	-
$\eta_{\boldsymbol{P_g}}(\%)$	100	100
$\eta_{\boldsymbol{Q_g}}(\%)$	100	100
$\eta_{\boldsymbol{S_l}}(\%)$	100	100
$\eta_{\boldsymbol{\theta_l}}(\%)$	100	100
$\eta_{\boldsymbol{P_d}}(\%)$	>99	>99
$\eta_{\boldsymbol{Q_d}}(\%)$	>99	**86**
η_{sp}	1658	1663

(loads) and output (voltage magnitudes and angles of all buses) but different loss functions. The dataset consists of samples from two load-solution mappings with an average of 30% difference in objectives. For each load input, the numbers of the low-cost and high-cost solutions are equal.

Table. III indicates that both two approaches have good performances in feasibility (i.e., all constraints are satisfied) and computational speedup (i.e., up to three orders of magnitude). Although DeepOPF-NGT has a bit larger optimality loss than DeepOPF-V, it has much higher load satisfaction ratios. More than 99.5% active and reactive loads are satisfied by DeepOPF-NGT, while only 86% reactive loads are satisfied by DeepOPF-V. This suggests that DeepOPF-NGT has better performance than DeepOPF-V when there are multiple load-solution mappings embedded in the dataset.

D. Performance of Extended DeepOPF-NGT

To explore the benefits brought by a few ground truths, we compare DeepOPF-NGT with its extended version in the modified IEEE 118/300-bus systems. We only use 300 training samples without ground truths and 0∼250 samples with ground truths to explore how ground truths help to improve the performance of DeepOPF-NGT. The results are summarized in Table. IV, where N_{label} is the number of samples with ground truths, and t_{train} is the training time. Note that better performances can be achieved if there are more training samples without ground truths.

Table. IV shows that the well-trained DNN models achieve good performance in the 118-bus and 300-bus systems with 8e3 and 1e4 epochs, respectively. They both have decent computational speedup up to three orders of magnitude and minor optimality loss less than 0.35%. Besides, they provide solutions with good feasibility. Almost all constraints are satisfied with high satisfaction ratios over 99.30%, and the load satisfaction ratios are all over 99.50%. The optimality loss in the 300-bus system is negative, which means DeepOPG-NGT finds a solution with a smaller objective than the ground truth. One possible reason is that DeepOPF-NGT tries to find the solution with the smallest objective during the training, so it may find a better solution than the solver MIPS which may give a locally optimal solution. Another reason is that some loads are not fully satisfied, resulting in less power generation.

Fig. 5 shows the comparison results of the predicted solutions and their ground truths in the 300-bus system. Due to the space limitations, we only present 10 samples selected uniformly at random in the test dataset. Fig. 5(a) shows that the predicted objective is close to its ground truth, despite the small deviations between the predicted power generations and their ground truths in Fig. 5(b). Fig. 5(c) shows the predicted and actual loads are very close to each other.

When the number of epochs decreases from 8000 to 3000, t_{train} decreases but the performance of DeepOPF-NGT becomes worse. For example, in the 118-bus system, the optimality loss increases from 0.26% to 4.04%, and the constraint satisfaction ratios of the active and reactive power generations decrease from 100% to 98.09% and 99.97%, respectively. Nevertheless, all performances tend to be improved with N_{label} increasing (see Table. IV). The results in the 300-bus system are shown in Fig. 6, showing an obvious tendency that the ground truth can help improve DeepOPF-NGT.

Note that when there are more samples with ground truths, more training time will be required in the pre-training process in Algorithm 2. Thus, t_{train} of the extended DeepOPF-NGT increases with N_{label} increasing. Despite this, compared with t_{train} of the well-trained DeepOPF-NGT, the overall training time is still reduced significantly by nearly half in the 118-bus and 300-bus systems.

E. Performance of Proposed Adaptive Learning Rate Algorithm

We compare the proposed adaptive learning rate algorithm with the following two methods in the modified IEEE 118-bus system to demonstrate its effectiveness: 1) fixed coefficients for different loss terms, a common method in existing studies; 2) learning rate annealing algorithm in [19]. Table. V shows that the proposed method outperforms the other two methods, with less than 0.1% optimality loss and more than 99% load satisfaction ratio, while the algorithm in [19] has the worst performance, with more than 4.5% optimality loss and less than 93% load satisfaction ratio. Since the loss function is updated dynamically during training, using fixed coefficients can lead to dominant loss terms that may cause unbalanced back-propagated gradients and bias the training towards ignoring the contributions of small loss terms [19]. Regarding the learning rate annealing algorithm [19], it may have exploding gradients due to unbounded coefficients for different loss terms.

The above results are further verified in Fig. 7. At the final (3000-th) training epoch, the gradient magnitudes of different loss terms are more similar to each other in the proposed method than in the other two methods. Hence, the proposed algorithm has the most balanced gradients between different loss terms. In the fixed coefficient method (see Fig. 7 (b)), the gradient magnitudes of $\mathcal{L}_d$ dominate those of $\mathcal{L}_{obj}$ and the loss terms in $\mathcal{L}_{cons}$ (denoted as $\widetilde{\mathcal{L}}_{cons}$), resulting in more optimality loss and constraint violations. In the learning rate annealing algorithm [19] (see Fig. 7 (c)), $\mathcal{L}_{cons}$ has dominant gradient magnitudes, which can lead to large optimality loss and a small load satisfaction ratio, as shown in Table. V.

VI. CONCLUSION

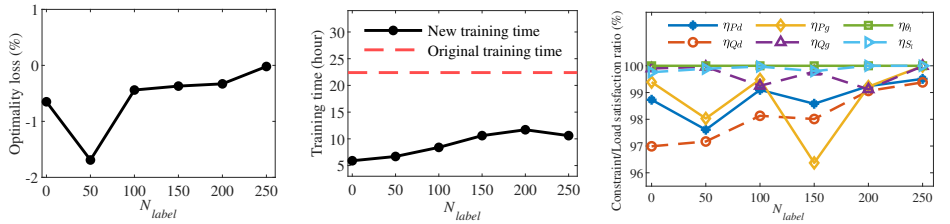
We propose an *unsupervised* learning approach called DeepOPF-NGT to solve AC-OPF problems efficiently without ground truths. It predicts bus voltages and then reconstructs

TABLE IV
PERFORMANCE OF EXTENDED DEEPOPF-NGT IN THE MODIFIED IEEE 118-BUS AND 300-BUS SYSTEMS.

Metric	IEEE 118-bus system							IEEE 300-bus system						
	DeepOPF-NGT		Extended DeepOPF-NGT					DeepOPF-NGT		Extended DeepOPF-NGT				
N_{label}	0		50	100	150	200	250	0		50	100	150	200	250
Epoch	8000	3000	3000					10000	3000	3000				
$\eta_{opt}(\%)$	**0.26**	**4.04**	1.08	1.71	2.24	1.22	**0.63**	**0.33**	**-0.65**	-1.69	-0.44	-0.37	-0.33	**-0.02**
$\eta_{\boldsymbol{P_g}}(\%)$	**100.00**	**98.09**	99.70	100.00	96.46	99.56	**99.98**	99.75	99.39	98.03	99.49	96.37	99.22	99.99
$\eta_{\boldsymbol{Q_g}}(\%)$	100.00	99.97	99.97	100.00	99.98	100.00	100.00	99.90	99.96	99.25	99.78	99.12	99.99	99.96
$\eta_{\boldsymbol{S_l}}(\%)$	99.33	99.55	99.60	99.71	99.57	99.57	99.43	99.92	99.76	99.89	99.97	99.80	100.00	100.00
$\eta_{\boldsymbol{\theta_l}}(\%)$	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
$\eta_{\boldsymbol{P_d}}(\%)$	99.53	99.70	99.65	99.39	99.60	99.52	99.39	**99.79**	**98.73**	97.61	99.10	98.58	99.24	**99.51**
$\eta_{\boldsymbol{Q_d}}(\%)$	99.91	99.51	99.94	99.90	98.66	99.92	99.96	99.44	99.99	97.17	98.13	98.01	99.06	99.38
η_{sp}	1128	1236	1232	1197	1252	1109	1254	1048	1042	934	947	977	1047	1058
t_{train}(hour)	**1.88**	**0.65**	0.72	0.92	1.07	1.08	**1.17**	**22.40**	**5.87**	6.71	8.43	10.63	11.74	**10.63**

(a) Objective of AC-OPF (b) Power generation at bus 98 (c) Active and reactive loads at buses 1 and 5

Fig. 5. Experimental results of DeepOPF-NGT in the modified IEEE 300-bus system.



(a) Optimality loss (b) Training time (c) Constraint and load satisfaction ratios

Fig. 6. The impact of the number of samples with ground truths on various performance of DeepOPF-NGT in the modified IEEE 300-bus system.

TABLE V
COMPARISON RESULTS OF DIFFERENT LEARNING RATE ALGORITHMS.

Metric	Proposed	Fixed coefficient	Reference [19]
$\eta_{opt}(\%)$	$<$ **0.1**	$>$ **2.5**	$>$ **4.5**
$\eta_{\boldsymbol{V}}(\%)$	-	-	-
$\eta_{\boldsymbol{P_g}}(\%)$	100	99.98	100
$\eta_{\boldsymbol{Q_g}}(\%)$	100	100	100
$\eta_{\boldsymbol{S_l}}(\%)$	99.43	99.44	99.46
$\eta_{\boldsymbol{\theta_l}}(\%)$	100	100	100
$\eta_{\boldsymbol{P_d}}(\%)$	99.30	**97.23**	**92.72**
$\eta_{\boldsymbol{Q_d}}(\%)$	99.93	**98.79**	**96.58**

all remaining variables by power flow equations. Distinct from existing approaches that learn the load-solution mapping embedded in the training dataset, DeepOPF-NGT directly learns a valid load-solution mapping without ground truths. A loss function consisting of the objective and penalties for constraint violations of the AC-OPF problem is properly designed to guide the DNN training using mini-batch stochastic gradient descent algorithm. An adaptive learning rate algorithm is devised to handle unbalanced gradient pathologies. We have also derived the first condition (to our best knowledge) for *unsupervised* learning to learn a valid load-solution mapping. Simulation results on the modified 118-bus system verify that DeepOPF-NGT has comparable performance with the state-of-the-art supervised-learning approaches. It provides feasible solutions with minor optimality loss (less than 0.4% in the 118-bus and 300-bus systems) and a decent computational speedup (up to three orders of magnitude faster than conventional solver MIPS). Besides, the results in the modified IEEE 39-bus system indicate that DeepOPF-NGT achieves better performance than the *supervised* learning approach when there are multiple load-solution mappings embedded in the dataset. Moreover, the results in the modified IEEE 118-bus and 300-bus systems

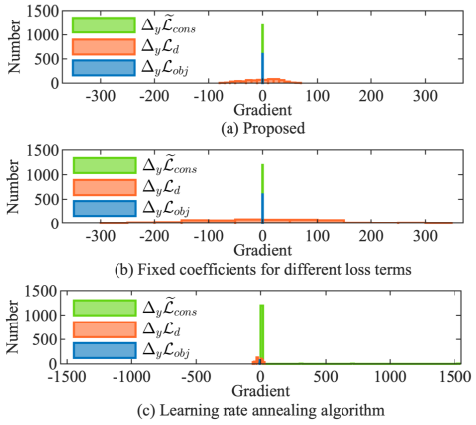


Fig. 7. The gradients of different loss terms at the 3000-th training epoch in the modified IEEE 118-bus system, where $\Delta_{\boldsymbol{y}}\widetilde{\mathcal{L}}_{cons}$ contains $k_g\Delta_{\boldsymbol{y}}\mathcal{L}_g$, $k_{S_l}\Delta_{\boldsymbol{y}}\mathcal{L}_{S_l}$ and $k_{\theta_l}\Delta_{\boldsymbol{y}}\mathcal{L}_{\theta_l}$.

indicate that a few ground truths can help reduce the training time and enhance the performance of DeepOPF-NGT. The effectiveness of the proposed adaptive learning rate method is also validated in the IEEE 118-bus system.

An interesting future direction is to design a neural network architecture to balance the interplay between different terms in the loss function and explore more efficient training algorithms.

APPENDIX

A. Proof of Lemma 1

Proof: By A1, we have $\left(\psi(\boldsymbol{y}_i)-\psi(\boldsymbol{y}_j)\right)^T(\boldsymbol{y}_i-\boldsymbol{y}_j)\geq L_\psi\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2^2$. Since $\left(\psi(\boldsymbol{y}_i)-\psi(\boldsymbol{y}_j)\right)^T(\boldsymbol{y}_i-\boldsymbol{y}_j)\leq\|\psi(\boldsymbol{y}_i)-\psi(\boldsymbol{y}_j)\|_2\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2$, we can obtain (10). ■

B. Proof of Theorem 2

Assume ξ_{nn} is not a valid mapping. We will show by contradiction that this assumption violates C1–C3 in Theorem 2. For $\boldsymbol{x}\in\mathcal{X}_{ij}^{\varepsilon}$, we do not differentiate the target mappings because they are too close to each other. Condition C2 ensures that ξ_{nn} has good optimality and feasibility performances. Thus, we will focus on the remaining region $\boldsymbol{x}\in\mathcal{X}\backslash\mathcal{X}_{ij}^{\varepsilon}$.

Assume there exists $\boldsymbol{x}\in\mathcal{X}\backslash\mathcal{X}_{ij}^{\varepsilon}$ such that ξ_{nn} consists of two target mappings ξ_i, $\xi_j\in\boldsymbol{\xi}$. By Definition 3 and C2, $\|\psi(\xi_{nn}(\boldsymbol{x}))-\psi(\xi_i(\boldsymbol{x}))\|_2\leq\delta$, and $\|\psi(\xi_{nn}(\boldsymbol{x}))-\psi(\xi_j(\boldsymbol{x}))\|_2\leq\delta$. By Lemma 1, $\|\xi_{nn}(\boldsymbol{x})-\xi_i(\boldsymbol{x})\|_2\leq\delta/L_\psi$, and $\|\xi_{nn}(\boldsymbol{x})-\xi_j(\boldsymbol{x})\|_2\leq\delta/L_\psi$. Then, we have

$$\|\xi_{nn}(\boldsymbol{x})-\xi_i(\boldsymbol{x})\|_2+\|\xi_{nn}(\boldsymbol{x})-\xi_j(\boldsymbol{x})\|_2\leq 2\delta/L_\psi \quad (13)$$

Since $\|\xi_i(\boldsymbol{x})-\xi_j(\boldsymbol{x})\|_2=\|\xi_i(\boldsymbol{x})-\xi_{nn}(\boldsymbol{x})+\xi_{nn}(\boldsymbol{x})-\xi_j(\boldsymbol{x})\|_2\leq\|\xi_i(\boldsymbol{x})-\xi_{nn}(\boldsymbol{x})\|_2+\|\xi_{nn}(\boldsymbol{x})-\xi_j(\boldsymbol{x})\|_2$, we have the following inequation by C1:

$$\|\xi_i(\boldsymbol{x})-\xi_{nn}(\boldsymbol{x})\|_2+\|\xi_{nn}(\boldsymbol{x})-\xi_j(\boldsymbol{x})\|_2\geq\varepsilon \quad (14)$$

According to (13)–(14), we obtain $\delta<\varepsilon L_\psi/2$ which contradicts C1. Thus, the assumption does not hold, and ξ_{nn} is a valid mapping.

REFERENCES

[1] A. Schecter and R. P. O'Neill, "Exploration of the ACOPF feasible region for the standard IEEE test set," *FERC Staff Technical Paper*, 2013.

[2] F. Capitanescu, "Critical review of recent advances and further developments needed in AC optimal power flow," *Electr. Power Syst. Res.*, vol. 136, pp. 57–68, 2016.

[3] B. Huang and J. Wang, "Applications of physics-informed neural networks in power systems-A review," *IEEE Trans. Power Syst.*, 2022.

[4] 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, 2022.

[5] W. Dong, Z. Xie, G. Kestor, and D. Li, "Smart-PGSim: Using neural network to accelerate AC-OPF power grid simulation," in *Proc SC20: Int. Conf. High Perform. Comput., Netw., Storage Anal.,*. IEEE, 2020, pp. 1–15.

[6] Y. Chen and B. Zhang, "Learning to solve network flow problems via neural decoding," *arXiv preprint arXiv:2002.04091*, 2020.

[7] F. Hasan, A. Kargarian, and J. Mohammadi, "Hybrid learning aided inactive constraints filtering algorithm to enhance AC OPF solution time," *IEEE Trans. Ind. Applicat.*, vol. 57, no. 2, pp. 1325–1334, 2021.

[8] X. Pan, T. Zhao, and M. Chen, "DeepOPF: Deep neural network for DC optimal power flow," in *Proc. IEEE Int. Conf. Smart Grid Commun.*, Beijing, China, 2019.

[9] 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, 2021.

[10] A. S. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," in *Proc. IEEE Int. Conf. Smart Grid Commun.*, 2020.

[11] K. Baker, "Emulating AC OPF solvers for obtaining sub-second feasible, near-optimal solutions," *arXiv preprint arXiv:2012.10031*, 2020.

[12] M. Singh, V. Kekatos, and G. B. Giannakis, "Learning to Solve the AC-OPF using Sensitivity-Informed Deep Neural Networks," *IEEE Trans. Power Syst.*, pp. 1–1, 2021.

[13] M. Chatzos, F. Fioretto, T. W. Mak, and P. Van Hentenryck, "High-fidelity machine learning approximations of large-scale optimal power flow," *arXiv preprint arXiv:2006.16356*, 2020.

[14] K. Lehmann, A. Grastien, and P. Van Hentenryck, "AC-feasibility on tree networks is NP-hard," *IEEE Trans. Power Syst.*, vol. 31, no. 1, pp. 798–801, 2015.

[15] H.-D. Chiang and T. Wang, "On the existence of and lower bounds for the number of optimal power flow solutions," *IEEE Trans. Power Syst.*, vol. 34, no. 2, pp. 1116–1126, 2018.

[16] J. Kotary, F. Fioretto, and P. Van Hentenryck, "Learning hard optimization problems: A data generation perspective," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 24 981–24 992, 2021.

[17] Z. Yan and Y. Xu, "Real-time optimal power flow: A lagrangian based deep reinforcement learning approach," *IEEE Trans. Power Syst.*, vol. 35, no. 4, pp. 3270–3273, 2020.

[18] J. H. Woo, L. Wu, J.-B. Park, and J. H. Roh, "Real-time optimal power flow using twin delayed deep deterministic policy gradient algorithm," *IEEE Access*, vol. 8, pp. 213 611–213 618, 2020.

[19] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," *SIAM J. Sci. Comput.*, vol. 43, no. 5, pp. A3055–A3081, 2021.

[20] P. L. Donti, D. Rolnick, and J. Z. Kolter, "DC3: A learning method for optimization with hard constraints," *Proc. Int. Conf. Learn. Representations*, 2021.

[21] J. Wang and P. Srikantha, "Fast optimal power flow with guarantees via an unsupervised generative model," *IEEE Trans. Power Syst.*, pp. 1–12, 2022.

[22] D. Owerko, F. Gama, and A. Ribeiro, "Unsupervised optimal power flow using graph neural networks," *arXiv preprint arXiv:2210.09277*, 2022.

[23] W. A. Bukhsh, A. Grothey, K. I. M. McKinnon, and P. A. Trodden, "Local solutions of the optimal power flow problem," *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 4780–4788, 2013.

[24] R. Salgado, C. Moyano, and A. Medeiros, "Reviewing strategies for active power transmission loss allocation in power pools," *Int. J. Electr. Power Energy Syst.*, vol. 26, no. 2, pp. 81–90, 2004.

[25] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited on*, vol. 14, no. 8, p. 2, 2012.

[26] E. K. Ryu and S. Boyd, "Primer on monotone operator methods," *Appl. comput. math*, vol. 15, no. 1, pp. 3–43, 2016.

[27] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, 2011.
[28] Y. Tang, K. Dvijotham, and S. Low, "Real-time optimal power flow," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2963–2973, 2017.