2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)

# DeepOPF: Deep Neural Network for DC Optimal Power Flow

Xiang Pan, Tianyu Zhao, and Minghua Chen
Department of Information Engineering, The Chinese University of Hong Kong

***Abstract*—We develop DeepOPF as a Deep Neural Network (DNN) based approach for solving direct current optimal power flow (DC-OPF) problems. DeepOPF is inspired by the observation that solving DC-OPF for a given power network is equivalent to characterizing a high-dimensional mapping between the load inputs and the dispatch and transmission decisions. We construct and train a DNN model to learn such mapping, then we apply it to obtain optimized operating decisions upon arbitrary load inputs. We adopt uniform sampling to address the over-fitting problem common in generic DNN approaches. We leverage on a useful structure in DC-OPF to significantly reduce the mapping dimension, subsequently cutting down the size of our DNN model and the amount of training data/time needed. We also design a post-processing procedure to ensure the feasibility of the obtained solution. Simulation results of IEEE test cases show that DeepOPF always generates feasible solutions with negligible optimality loss, while speeding up the computing time by *two orders of magnitude* as compared to conventional approaches implemented in a state-of-the-art solver.**

## I. Introduction

As the most widely-used and mature model in deep learning, Deep Neural Network (DNN) [1] demonstrates superb performance in complex engineering tasks. The capability of approximating continuous mappings and the desirable scalability make DNN a favorable choice in the arsenal of solving large-scale optimization and decision problems. In this paper, we apply deep learning to power systems and develop a DNN approach for solving the essential optimal power flow (OPF) problem in power system operation.

The OPF problem, first posed by Carpentier in 1962 in [2], is to minimize an objective function, such as the cost of power generation, subject to all physical, operational, and technical constraints, by optimizing the dispatch and transmission decisions. These constraints include Kirchhoff's laws, operating limits of generators, voltage levels, and loading limits of transmission lines [3]. The OPF problem is central to power system operations as it underpins various applications including economic dispatch, unit commitment, stability and reliability assessment, and demand response. While OPF with a full AC power flow formulation (AC-OPF) is most accurate, it is a non-convex problem and its complexity obscure practicability. Meanwhile, based on linearized power flows, DC-OPF is a convex problem admitting a wide variety of applications, including electricity market clearing and power transmission management. See e.g., [4], [5] for a survey.

Our DNN approach is inspired by the following observations on the characteristics of OPF and its application in practice.

- Given a power network, solving the OPF problem is equivalent to depicting a high-dimensional mapping between load inputs and optimized dispatch and transmission decisions.
- In practice, the OPF problem is usually solved repeatedly for the same power network, e.g., every five minutes by CAISO, with different load inputs at different time epochs.

As such, it is conceivable to leverage the universal approximation capability of deep feed-forward neural networks [6], [7] to learn such mapping for a given power network, and then apply the mapping to obtain operating decisions upon giving load inputs (e.g., once every five minutes).

In this paper, we develop DeepOPF as a DNN approach for solving DC-OPF problems, which integrates the structure of the DC-OPF problem into the design of the DNN model and the loss function. The framework also includes a pre-processing procedure to calibrate the inputs to improve DNN training efficiency and a post-processing procedure to ensure the feasibility of the solutions obtained from the DNN model. As compared to conventional approaches based on interior-point methods [8], DeepOPF excels in (i) reducing computing time and (ii) scaling well with the problem size. These salient features are particularly appealing for solving the (large-scale) security-constrained DC-OPF problem, which is central to secure power system operation with contingency in consideration. Note that the complexity of constructing and training a DNN model is minor if amortized over the many DC-OPF instances (e.g., one per every five minutes) that can be solved using the same model. Simulation results of IEEE test cases show that DeepOPF always generates feasible solutions with negligible optimality loss, while speeding up the computing time by *two orders of magnitude* as compared to conventional approaches. As the initial work on applying the deep neural network to solving OPF problems, we focus on the simple setting of DC-OPF to illustrate the idea and highlight the potential. The DeepOPF approach is applicable to more general settings, including the large-scale security-constrained OPF [9] and non-convex AC-OPF problems, which we leave for future studies.

## II. Related Work

Due to the space limitation, we focus on the most related works in this section; a more comprehensive discussion is presented in our technical report [10]. Existing research mainly focuses on two categories of methods for solving the OPF problem. One is the methods based on numerical iteration algorithms. The OPF problem to be solved can be first approximated as an optimization problem like quadratic programming [11], or linear programming [12], and the numerical iteration

978-1-5386-8099-5/19/$31.00 ©2019 IEEE

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on May 03,2023 at 07:44:48 UTC from IEEE Xplore. Restrictions apply.

solvers like interior-point methods [13], were applied to obtain the optimal solutions. However, the time complexity of these numerical-iteration based algorithms may be substantial. Usually the computation time increases when the scale of transmission power system becomes large.

The other category is learning-based methods. Some work focused on finding an alternative approach to solve the OPF problem [14], yet these methods may not always generate feasible solutions. Other work investigated how to integrate the learning techniques into the conventional algorithm to speed up the process of solving OPF problems by for instance proper initialization. However, the resulting heuristic schemes are still iteration based and may still incur a significant amount of running time for large-scale instances. Apart from that, [7] presented an approach to solve the constrained finite-time optimal control problem based on deep learning, which mainly focuses on achieves high accuracy with fewer memory requirements. Despite such similarity, however, this paper mainly focuses on how to solve the OPF problem faster by levering the deep learning technique.

## III. DC-OPF Problem

In the DC-OPF problem, there are two types of variables, i.e., the generator outputs and the power phase angle of transmission lines. The problem is to minimize the total generation cost subject to the generator operation limits, the power balance equation, and the transmission line capacity constraints [15] as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{N_{\text{gen}}} C_i\left(P_{Gi}\right) \\ \text{s.t.} \quad & \begin{cases} P_{Gi}^{\min} \le P_{Gi} \le P_{Gi}^{\max},\ i=1,2,...,N_{\text{bus}} \\ \mathbf{B}\cdot\theta = \mathbf{P_G} - \mathbf{P_D} \\ \frac{1}{x_{ij}}\left(\theta_i-\theta_j\right) \le P_{ij}^{max},\ i,j=1,2,...,N_{\text{bus}} \end{cases} \end{aligned} \tag{1}$$

where $N_{\text{bus}}$ is the number of buses and $N_{\text{gen}}$ is the number of generators. $P_{Gi}$ is the power output of the generator in the $i$th bus. $P_{Gi}^{\min}$ and $P_{Gi}^{\max}$ are the output limits of generators in the $i$th bus, respectively. If there is no generator in the $i$the bus, $P_{Gi}^{\min}$ and $P_{Gi}^{\max}$ are set to be 0. $\mathbf{B}$ is an $N_{\text{bus}} \times N_{\text{bus}}$ admittance matrix. If the $i$the bus and the $j$the bus are adjacent buses, then the corresponding element in matrix $\mathbf{B}$ is the reciprocal of the line reactance $x_{ij}$, Otherwise the corresponding element is 0. $\mathbf{P_G}$ is the bus power generation vector and $\mathbf{P_D}$ is the bus consumption vector. $\theta$ is the phase angles vector. $\theta_i$ and $\theta_j$ are the phase angles at the $i$th bus and the $j$th bus, respectively. $\frac{1}{x_{ij}}\left(\theta_i-\theta_j\right)$ represents the bus power injection from the $j$th bus and the $i$th bus. $P_{ij}^{\max}$ is the transmission limit from the $i$th bus to the $j$ bus. $C_i\left(P_{Gi}\right)$ is individual cost function for the generator in the $i$th bus. The cost function is derived from a heat rate curve, which gives the generator electric power output as a function of the thermal energy input rate times the fuel cost per thermal energy unit. It is commonly modeled as a quadratic function [16]:

$$C_i\left(P_{Gi}\right) = \lambda_{1i}^2 P_{Gi} + \lambda_{2i} P_{Gi} + \lambda_{3i}, \tag{2}$$

where $\lambda_{1i}$, $\lambda_{2i}$, and $\lambda_{3i}$ are the model parameters. The parameters of the cost function can be obtained from measured data of the heat rate curve [15].

## IV. DeepOPF for Solving DC-OPF

### A. Overview of DeepOPF

The flowchart of the proposed framework is depicted in Fig. 1. First, we collect the training data and perform pre-processing. Specifically, we apply a uniform sampling method to generate the load $\mathbf{P_D}$. We then obtain optimal solution $\mathbf{P_G}$ and $\theta$ for the corresponding DC-OPF problems as the ground-truth, by using a state-of-the-art solver [17]. After that, the training data will be normalized during the pre-possessing. The details are in Sec. IV-B. Note that uniform sampling can address the over-fitting issue in generic DNN approaches.

Second, we study an equivalent formation of DC-OPF to use a scaling factor vector $\hat{\alpha} \in [0,1]$ to represent the active power $\mathbf{P_G}$, normalizing the output value ranges which are known to facilitate training efficiency. Furthermore, we leverage on the fact that the admittance matrix (after removing the entries corresponding to the slack bus) is full rank to represent $\theta$ by $\mathbf{P_G}$. Thus, it suffices to learn only the mapping from load inputs to the active powers instead of to both the active powers and the phase angles as in generic DNN approaches. This way, we can reduce the size of the DNN model and the amount of training data/time needed. The details are in Sec. IV-C.

Third, we build a DNN model with $N_{\text{hid}}$ hidden layers with $N_{\text{neu}}$ on each layer based on the scale of the power system to solve the fitting problem of the scaling factor vector $\hat{\alpha}$. We train the DNN model by applying the data-driven stochastic gradient descent optimization algorithm to minimize a carefully-chosen loss function designed for DC-OPF problems. The details are in Sec. IV-D.

Fourth, we integrate post-processing into the DeepOPF to guarantee the feasibility of the solutions obtained by the DNN model. If the DNN model generates infeasible solutions, DeepOPF will project the solution into the feasible region and return a feasible solution. The details are in Sec. IV-E.

### B. load sampling and pre-processing

As mentioned in Section I, we assume the load on each bus varies within a specific range around the default value independently. The load data is sampled within $[(1-x)*P_d, (1+x)*P_d]$ ($P_d$ is the default power load at individual bus $d$ and $x$ is the percentage of sample range like $10\%$) uniformly at random. It is then fed into the traditional DC-OPF solver to generate optimal solutions. As the magnitude for each dimension of the input and output may be different, each dimension of training data will be normalized with the standard variance and mean of the corresponding dimension.

### C. Linear transformation and mapping dimension reduction

There are inequality constraints related to $P_{Gi}$, we first reformulated them through linear scaling as:

$$P_{Gi} = \alpha_i \cdot \left(P_{Gi}^{\max} - P_{Gi}^{\min}\right),\ \alpha_i \in [0,1], i = 1,...,N_{\text{gen}}, \tag{3}$$

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on May 03,2023 at 07:44:48 UTC from IEEE Xplore. Restrictions apply.

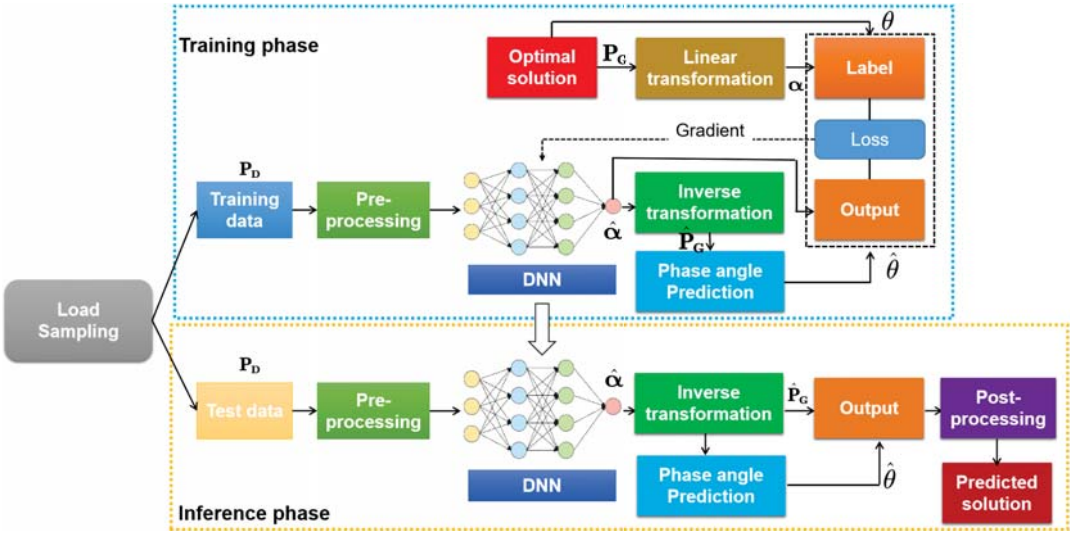


Fig. 1. The framework of DeepOPF.

where we recall that $N_{\text{gen}}$ is the number of generators. Thus, instead of predicting the generated power, we can predict the scaling factor $\alpha_i$ and obtain the value of $P_{Gi}$. There are two advantages to this approach. First, the range of $\alpha_i$ naturally matches the output range of the DNN restricted by the sigmoid function [1]. Thus, it automatically prevents the recovery of the predictions from violating the inequality constraints. Second, the prediction range becomes smaller, which makes it easier for the DNN model to learn the mapping between the load and $P_{Gi}$. It should be noted that one bus is set as the slack bus and used for balancing the mismatch between the total load and supply, thus the $P_{Gi}$ of the slack bus is obtained by subtracting the output of the other buses from the total load.

Since the phase angle, $\theta_i$ is the state variable depends on the decision variable $P_{Gi}$, and there exists a linear relationship between them (the second equality constraint in the DC-OPF problem). Thus, after obtaining $P_{Gi}, i = 1, ..., N_{\text{gen}}$, we can have bus power generation vector $\mathbf{P_G}$, and can compute $\theta$ accordingly. Suppose we obtain the $(N_{\text{bus}} - 1) \times (N_{\text{bus}} - 1)$ matrix, $\tilde{\mathbf{B}}$ by eliminating corresponding row and column of the $n \times n$ admittance matrix $\mathbf{B}$ to the slack bus whose phase angle is usually set to zero. The following lemma states a useful property of the admittance matrix B pretty well-known in the literature; see e.g., [15], [18].

*Lemma 1:* The matrix $\tilde{\mathbf{B}}$ is a full-rank matrix.

According to Lemma 1, we can express the phase angles of all bus except the phase angle of the slack bus as follows:

$$\tilde{\theta} = \left(\tilde{\mathbf{B}}\right)^{-1} \left(\tilde{\mathbf{P}}_{\mathbf{G}} - \tilde{\mathbf{P}}_{\mathbf{D}}\right), \tag{4}$$

where $\tilde{\mathbf{P}}_{\mathbf{G}}$ and $\tilde{\mathbf{P}}_{\mathbf{D}}$ stand for the $(N_{\text{bus}} - 1)$-dimension output and load vectors for buses excluding the slack bus, respectively. As the phase angle of the slack bus is fixed, the DNN model does not need to predict it. There are mainly two advantages of this transformation. On one hand, we use the property of the admittance matrix to reduce the number of variables to predict, which further reduces the size of our DNN model and the amount of training data/time needed. On the other hand, the linear transformation in (4) makes it convenient to model error related to $\theta$ concerning $\hat{\mathbf{P}}_{\mathbf{G}}$ in the loss function.

### D. The DNN model

The core of DeepOPF is the DNN model applied to approximate the mapping between the load and power output of the generators. The DNN model is established based on the multi-layer feed-forward neural network structure, which is defined as:

$$\begin{aligned} h_0 &= \tilde{\mathbf{P}}_{\mathbf{D}}, \\ h_i &= \sigma\left(W_i h_{i-1} + b_{i-1}\right), \\ \hat{\alpha} &= \sigma'\left(w_o h_L + b_o\right), \end{aligned}$$

where $h_0$ denotes the input vector of the network, $h_i$ is the output vector of the $i$th hidden layer, $h_L$ is the output vector, and $\hat{\alpha}$ is the generated scaling factor vector for the generators. Matrices $W_i$, biases vectors $b_i$, and activation functions $\sigma(\cdot)$ and $\sigma'(\cdot)$ are subject to the DNN design.

*1) The architecture:* In the DNN model, $h_0$ represents the normalized load data, which is the inputs of the network. After that, features are learned from the input vector $h_0$ by several fully connected hidden layers. The $i$th hidden layer models the interactions between features by introducing a connection weight matrix $W_i$ and a bias vector $b_i$. Activation function $\sigma(\cdot)$ further introduces non-linearity into the hidden layers. In our DNN model, we adopt the widely-used Rectified Linear Unit (ReLU) as the activation function of the hidden layers. At last, the Sigmoid function $\sigma'(x) = \frac{1}{1+e^{-x}}$ is applied as activation function of the output layer to project the outputs of the network to $(0, 1)$. For different power networks (as IEEE test cases), we tailor the DNN model by educated guesses and iterative tuning, which is by far the common practice in generic DNN approaches in various engineering domains.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on May 03,2023 at 07:44:48 UTC from IEEE Xplore. Restrictions apply.

Details of designing the corresponding DNN can be found in our technical report [10].

*2) The loss function:* For each item in the training data set, the loss function consists of two parts: the difference between the generated solution and the reference solution obtained from solvers and the value of the penalty function upon solutions being infeasible. Since there exists a linear correspondence between $\mathbf{P_G}$ and $\theta$, there is no need to introduce the loss term of the phase angles. The difference between the generated solution and the actual solution of $P_{Gi}$ is expressed by the sum of mean square error between each element in the generated scaling factors $\hat{\alpha}_i$ and the actual scaling factors $\alpha_i$ in optimal solutions:

$$\mathcal{L}_{PG} = \frac{1}{N_{\text{gen}}} \sum_{i=1}^{N_{\text{gen}}} (\hat{\alpha}_i - \alpha_i)^2, \tag{5}$$

where $N_{\text{bus}}$ represents the number of generators. For the inequality constraints related to the transmission on each line, we can represent these constraints by $\mathbf{P_G}$ as following:

$$-1 \leq \frac{1}{P_{ij}^{\max} \cdot x_{ij}} \cdot (\theta_i - \theta_j) \leq 1, , \ i, j = 1, 2, ..., N_{\text{bus}}. \tag{6}$$

We first introduce an $n_a \times n$ matrix $\mathbf{A}$, where $n_a$ is the number of adjacent buses. Each row in $A$ corresponds to an adjacent bus pair. Given any the adjacent bus pair $(i, j)$, we assume the power flow is from the $i$th bus to the $j$th bus. Thus, the elements, $a_i$ and $a_j$, are the corresponding entries of the matrix $\mathbf{A}$ defined as:

$$a_i = \frac{1}{P_{ij}^{\max} \cdot x_{ij}} \text{ and } a_j = -\frac{1}{P_{ij}^{\max} \cdot x_{ij}}. \tag{7}$$

Based on (4) and (7), (6) can be expressed as:

$$-1 \leq \left(\mathbf{A}\hat{\theta}\right)_k \leq 1, k = 1, ..., n_a, \tag{8}$$

where $\left(\mathbf{A}\hat{\theta}\right)_k$ represents the $k$th element of $\mathbf{A}\hat{\theta}$. Thus, the phase angle vector $\hat{\theta}$ is obtained through $\mathbf{P_G}$ and $\bar{\theta}$. We can then calculate the penalty value for $\left(\mathbf{A}\hat{\theta}\right)_k$, and add the average penalty value into the loss function for training. The penalty term capturing the feasibility of the generated solutions can be expressed as:

$$\mathcal{L}_{pen} = \frac{1}{n_a} \sum_{k=1}^{n_a} p\left(\left(\mathbf{A}\hat{\theta}\right)_k\right), \tag{9}$$

where $n_a$ is the number of the adjacent buses, $p(\cdot)$ is the penalty function defined as $p(x) = x^2 - 1$, and $\left(\mathbf{A}\hat{\theta}\right)_k$ represents the $k$th element in the vector $\mathbf{A}\hat{\theta}$ ($\hat{\theta}$ is the generated phase angle vector by (4)). The total loss can be expressed as the weighted summation of the two parts:

$$\mathcal{L}_{total} = w_1 \cdot \mathcal{L}_{PG} + w_2 \cdot \mathcal{L}_{pen}, \tag{10}$$

where $w_1$ and $w_2$ are positive weighting factors, which are used to balance the influence of each term in the training phase. As the objective of the designed loss function is to find the relatively accurate $\mathbf{P_G}$, the first term is with the highest priority in the loss function as it has much more influence on the recovery of the $\mathbf{P_G}$. The other loss term, which is with respect to the penalty related to the transmission line, is regarded with lower priority in the training process.

*3) The training process:* In general, the training processing can be regarded as minimizing the average value of loss function with the given training data by tuning the parameters of the DNN model as follows:

$$\min_{W_i, b_i} \frac{1}{N_{\text{train}}} \sum_{k=1}^{N_{\text{train}}} \mathcal{L}_{total,k} \tag{11}$$

$N_{\text{train}}$ is the amount of training data and $\mathcal{L}_{total,k}$ is the loss of the $k$th item in the training.

We apply the widely-used optimization technique in the deep learning, stochastic gradient descent (SGD) [1], in the training stage, which is effective for the large-scale dataset and can economize on the computational cost at every iteration by choosing a subset of summation functions at every step.

### *E. Post-processing*

The proposed approach may encounter a situation in which the balanced amount of electricity exceeds the feasible capacity range of the slack bus. To address this issue and ensure the integrity of the algorithm, after the generated solution $\hat{\mathbf{P}}_\mathbf{G}$ is obtained, the approach needs to conduct post-processing to guarantee the feasibility of the solution.

As discussed before, the constraints in the DC-OPF problem are (closed) linear constraints. Thus, the post-processing can be regarded as to project the initial generated solution into the polyhedron that is the intersection of a finite number of closed linear constraints, which can be formulated as follows:

$$\min \ \|\hat{\mathbf{P}}_\mathbf{G} - \mathbf{u}\|^2 \ \text{ s.t. } \ \mathbf{u} \in C_1 \cap C_2 ... \cap C_d, \tag{12}$$

where the convex sets $C_1, C_2, ..., C_d$ denote the constraints in the DC-OPF problem. By solving the problem in (12), we can find the feasible solution closest to the generated solution $\hat{\mathbf{P}}_\mathbf{G}$. The reason that we want to find the feasible one that is closest to the pseudo-optimal solution lies in that, we want to provides the feasible one which has the smallest variation from $\hat{P}_G$, and expecting the value of the cost function is also close to the optimum. The above problem is a convex quadratic programming problem and can be solved by the fast dual proximal gradient algorithm proposed in [19], with a convergence rate of the primal sequence being of the order $\mathcal{O}(1/\tau)$, where $\tau$ is the iteration steps. With the post-processing, the proposed approach can guarantee the feasibility of the generated solution. The numerical experiments on IEEE test cases in Sec. V show that DeepOPF generates feasible solutions for all DC-OPF instances; thus in practice, maybe only few instances will involve the post-processing process.

### *F. Computational complexity*

As mentioned before, the computational complexity of the traditional iteration approach is related to the scale of the DC-OPF problem. For example, the computational complexity of

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on May 03,2023 at 07:44:48 UTC from IEEE Xplore. Restrictions apply.

TABLE I
PARAMETERS FOR STANDARD TEST CASES.

| Case | $N_{\text{bus}}$ | $N_{\text{gen}}$ | $N_{\text{load}}$ | $N_{\text{bran}}$ | $N_{\text{hid}}$ | $N_{\text{neu}}$ | $lr$ |
|---|---|---|---|---|---|---|---|
| 30 | 30 | 6 | 20 | 41 | 2 | 16 | 1e-3 |
| 7 | 57 | 7 | 42 | 80 | 4 | 32 | 1e-3 |
| 118 | 118 | 54 | 99 | 186 | 6 | 64 | 1e-3 |
| 300 | 300 | 69 | 199 | 411 | 6 | 128 | 1e-3 |

TABLE II
PERFORMANCE EVALUATION OF THE PROPOSED APPROACH ON IEEE STANDARD TEST CASES.

| Case | Feasibility rate. (%) | Ave. cost ($/hr) | | Time (millisecond) | | Speedup |
|---|---|---|---|---|---|---|
| | | DeepOPF | Ref. | DeepOPF | Ref. | |
| 30 | 100 | 589 | 588 | 0.19 | 21 | ×110 |
| 57 | 100 | 42750 | 42667 | 0.22 | 27 | ×122 |
| 118 | 100 | 133776 | 131311 | 0.29 | 44 | ×151 |
| 300 | 100 | 706602 | 706338 | 0.37 | 50 | ×135 |

* Feasibility rate represents the percentage of the feasible generated solution before post-processing.
* Time is the average running time of all test instances, which includes the computation time for the power generation and phase angle.

interior point method based approach for the convex quadratic programming is $\mathcal{O}\left(L^2n^4\right)$ measured as the number of arithmetic operations [20], where $L$ is the number of input bits and $n$ is the number of variables.

For our proposed approach, the computational complexity mainly consists of two parts: the calculation as the input data passing through the DNN model and the post-processing. For the post-processing, its computational complexity can be negligible in practical usage as the DNN model barely generate infeasible solutions. Thus, the computational complexity of the proposed approach is approximately determined by the calculation with respect to the DNN model. It can be evaluated by the scale of the network [21].

Recall that the input and the output of the DNN model in DeepOPF are $N_{\text{in}}$ and $N_{\text{out}}$ dimensions, respectively, and the DNN model has $N_{\text{hid}}$ hidden layers and each hidden layer has $N_n$ neurons. Once we finish training the DNN model, the complexity of generating solutions by using DeepOPF is characterized in the following proposition.

*Proposition 1:* The computational complexity (measured as the number of arithmetic operations) to generate scaling factors for $\hat{\mathbf{P}}_{\mathbf{G}}$ to the DC-OPF problem by using DeepOPF is given by

$$T = N_{\text{in}}N_{\text{n}} + (N_{\text{hid}} - 1)N_{\text{n}}^2 + N_{\text{out}}N_{\text{n}}, \tag{13}$$

which is $\mathcal{O}\left(N_{\text{hid}}N_{\text{n}}^2\right)$.

For example, as shown in Table I, for the parameters associated with IEEE 300 case, $T = 81920$. As for the given topology, the scale of the DNN model is fixed. It means the computational complexity $T$ is constant. After that, we can obtain the phase angle through simple linear operations.

## V. NUMERICAL EXPERIMENTS

### A. Experiment setup

*1) Simulation environment:* The experiments are conducted in Ubuntu 18.04 on the six-core (i5-8500@3.00G Hz) CPU workstation and 8GB RAM.

*2) Test case:* The proposed approach is tested with four IEEE standard cases [22]: the IEEE 30-bus power system, IEEE 57-bus power system, the IEEE 118-bus test system and the IEEE 300-bus system, respectively, which includes the small-scale, medium-scale and large-scale power system network for the DC-OPF problem. The related parameters for the test cases are shown in Table I.

*3) Training data:* In the training stage, the load data is sampled within $[90\%, 110\%]$ of the default load on each load uniformly at random. After that, the solution for the DC-OPF problem provided by the pypower [23] is regarded as ground-truth. Pypower is based on the traditional interior point method [17]. Taking the sampling range and different scale of the cases, the corresponding amount of training data for different cases is empirically determined in the simulation as follows: 10000 training data for IEEE Case30, 25000 training data for IEEE Case57 and IEEE Case118, 50000 training data for IEEE Case300. For each test case, the amount of test data is 10000.

*4) The implementation of the DNN model:* We design the DNN model based on Pytorch platform. Besides, the epoch is set to 200 and the batch size is 64. Based on the range of each loss obtained from some preliminary experiments, the value of weighting factors $w_1$ and $w_2$ are set to 1 and 0.00001 empirically. Meanwhile, other parameters like the number of hidden layers, the number of neurons in each layer and the learning rate for each test case are also shown in Table. I.

### B. Performance evaluation

The simulation results of the proposed approach for test cases are shown in Table II. We use the feasibility rate to represent the percentage of the feasible generated solution before post-processing. We can see from the Table. II the percentage of the feasible solution is 100% before post-processing, which indicates the proposed neural network model can keep the feasibility of the generated solution well without resorting to the post-process step. As the load of each node are assumed varying within a specific range, and the load for the training data is sampled in this range uniformly at random, the DNN model can find the mapping between the load in the specific range and the corresponding solution for the DC-OPF problem. It should be noted that the proposed neural network can learn the mapping for any varying range as long as there are enough sample data for training. Thus, the neural network model barely generates the infeasible solution, which means the proposed approach can mostly find a feasible solution through mapping. Even if it obtains the infeasible solution, the model can resort to the post-processing to adjust the infeasible generated solution into the feasible one. In addition, the difference between the cost with the generated solution and that of the reference solution is shown in the Table. II.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on May 03,2023 at 07:44:48 UTC from IEEE Xplore. Restrictions apply.

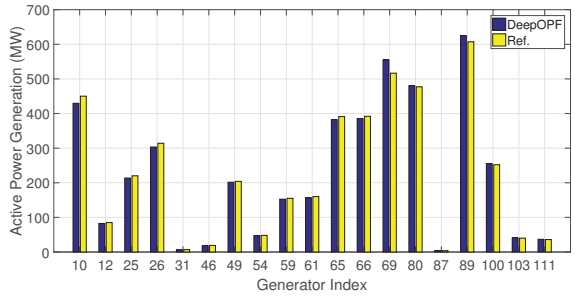


Fig. 2. The comparisons of several large generated solutions and the optimal solution for IEEE case-118.

The difference can be negligible, which means each dimension of the generated solution has high accuracy when compared to that of the optimal solution. To verify this, we show the comparisons between the generated solution and the optimal solution for several generators under IEEE case-118 with the given total load in Fig. 2 as an example. For better illustration, we show the comparisons of the generators with large output. We can observe from Fig. 2 that for prediction of the generators large output, the DeepOPF approach can not only describe the relative relation between each dimension in the optimal solution but also obtain prediction with a small difference. Usually, the total cost is mainly determined by the generators with larger output.

Apart from that, we can see that compared with the traditional DC-OPF solver, our DeepOPF approach can speed up the computing time by two order of magnitude. As mentioned before, given the topology of the power system, solving the OPF problem means to find the mapping between the load and decision variables of the generators. The proposed model can achieve high prediction accuracy much faster than the traditional iteration based solution. In addition, we provide the training time consumption (sec./epoch) as follows: case30 (0.3), case57 (3.6), case118 (10.4) and case300 (19.0). Recall that 200 epoches can achieve acceptable predicted solutions as shown in the simulation. As the OPF problem has to be solved frequently and repeatedly, the training time is negligible for the long-term DC-OPF application.

## VI. Conclusion

Solving DC-OPF optimally and efficiently is crucial for reliable and cost-effective power system operation. In this paper, we develop DeepOPF to generate feasible solutions for DC-OPF with negligible optimality loss. DeepOPF is inspired by the observations that solving DC-OPF for a given power network is equivalent to learning a high-dimensional mapping between the load inputs and the dispatch and transmission decisions. Simulation results show that DeepOPF scales well in the problem size and speed up the computing time by *two orders of magnitude* as compared to conventional of using modern convex solvers. These observations suggest two salient features particularly appealing for solving the large-scale security-constrained DC-OPF problems and the non-convex AC-OPF problems, which are both compelling future directions.

## References

[1] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*. MIT Press Cambridge, 2016, vol. 1.
[2] J. Carpentier, "Contribution to the economic dispatch problem," *Bulletin de la Societe Francoise des Electriciens*, vol. 3, no. 8, pp. 431–447, 1962.
[3] D. E. Johnson, J. R. Johnson, J. L. Hilburn, and P. D. Scott, *Electric Circuit Analysis*. Prentice Hall Englewood Cliffs, 1989, vol. 3.
[4] S. Frank, I. Steponavice, and S. Rebennack, "Optimal power flow: a bibliographic survey i," *Energy Systems*, vol. 3, no. 3, pp. 221–258, Sep 2012.
[5] ——, "Optimal power flow: a bibliographic survey ii," *Energy Systems*, vol. 3, no. 3, pp. 259–289, Sep 2012.
[6] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.
[7] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *arXiv preprint arXiv:1806.10644*, 2018.
[8] J. A. Momoh and J. Z. Zhu, "Improved interior point method for opf problems," *IEEE Transactions on Power Systems*, vol. 14, no. 3, pp. 1114–1120, Aug 1999.
[9] L. Liu, A. Khodaei, W. Yin, and Z. Han, "A distribute parallel approach for big data scale optimal power flow with security constraints," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct 2013, pp. 774–778.
[10] X. Pan, T. Zhao, and M. Chen, "Deepopf: Deep neural network for dc optimal power flow," *arXiv preprint arXiv:04479*, 2019.
[11] J. A. Momoh, "A generalized quadratic-based model for optimal power flow," in *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, Cambridge, MA, USA, Nov 1989, pp. 261–271.
[12] S. H. Low, "Convex relaxation of optimal power flowpart i: Formulations and equivalence," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 1, pp. 15–27, March 2014.
[13] A. A. Sousa and G. L. Torres, "Globally convergent optimal power flow by trust-region interior-point methods," in *2007 IEEE Lausanne Power Tech*, Lausanne, Switzerland, Jul 2007, pp. 1386–1391.
[14] Y. Ng, S. Misra, L. A. Roald, and S. Backhaus, "Statistical Learning For DC Optimal Power Flow," *arXiv preprint arXiv:1801.07809*, 2018.
[15] R. D. Christie, B. F. Wollenberg, and I. Wangensteen, "Transmission management in the deregulated environment," *Proceedings of the IEEE*, vol. 88, no. 2, pp. 170–195, Feb 2000.
[16] J. H. Park, Y. S. Kim, I. K. Eom, and K. Y. Lee, "Economic load dispatch for piecewise quadratic cost function using hopfield neural network," *IEEE Transactions on Power Systems*, vol. 8, no. 3, pp. 1030–1038, Aug 1993.
[17] H. Wang, C. E. Murillo-Sanchez, R. D. Zimmerman, and R. J. Thomas, "On computational issues of market-based optimal power flow," *IEEE Transactions on Power Systems*, vol. 22, no. 3, pp. 1185–1193, Aug 2007.
[18] A. M. Kettner and M. Paolone, "On the properties of the power systems nodal admittance matrix," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 1130–1131, Jan 2018.
[19] A. Beck and M. Teboulle, "A fast dual proximal gradient algorithm for convex minimization and applications," *Operations Research Letters*, vol. 42, no. 1, pp. 1–6, Jan 2014.
[20] Y. Ye and E. Tse, "An extension of karmarkar's projective algorithm for convex quadratic programming," *Mathematical Programming*, vol. 44, no. 1, pp. 157–179, May 1989.
[21] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, June 2015, pp. 5353–5360.
[22] "Power Systems Test Case Archive," 2018, http://labs.ece.uw.edu/pstca/.
[23] "pypower," 2018, https://pypi.org/project/PYPOWER/.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on May 03,2023 at 07:44:48 UTC from IEEE Xplore. Restrictions apply.