DeepOPF+: A Deep Neural Network Approach for DC Optimal Power Flow for Ensuring Feasibility

Tianyu Zhao*, Xiang Pan*, Minghua Chen†, Andreas Venzke‡, and Steven H. Low§
*Department of Information Engineering, The Chinese University of Hong Kong
†School of Data Science, City University of Hong Kong
‡Department of Electrical Engineering, Technical University of Denmark (DTU)
§Department of Computing and Mathematical Sciences, California Institute of Technology

***Abstract*—Deep Neural Networks approaches for the Optimal Power Flow (OPF) problem received considerable attention recently. A key challenge of these approaches lies in ensuring the feasibility of the predicted solutions to physical system constraints. Due to the inherent approximation errors, the solutions predicted by Deep Neural Networks (DNNs) may violate the operating constraints, e.g., the transmission line capacities, limiting their applicability in practice. To address this challenge, we develop DeepOPF+ as a DNN approach based on the so-called "preventive" framework. Specifically, we calibrate the generation and transmission line limits used in the DNN training, thereby anticipating approximation errors and ensuring that the resulting predicted solutions remain feasible. We theoretically characterize the calibration magnitude necessary for ensuring universal feasibility. Our DeepOPF+ approach improves over existing DNN-based schemes in that it ensures feasibility and achieves a consistent speed up performance in both light-load and heavy-load regimes. Detailed simulation results on a range of test instances show that the proposed DeepOPF+ generates 100% feasible solutions with minor optimality loss. Meanwhile, it achieves a computational speedup of *two orders of magnitude* compared to state-of-the-art solvers.**

NOMENCLATURE

Variable	**Definition**
$\mathcal{N}$	Set of buses, $N \triangleq \lvert\mathcal{N}\rvert$.
$\mathcal{G}$	Set of generators.
$\mathcal{D}$	Set of loads.
$\mathcal{E}$	Set of branches.
P_G	Power generation injection vector, $[P_{G_i}, i \in \mathcal{N}]$.
$P_G^{\min}$	Minimum generator output vector, $[P_{G_i}^{\min}, i \in \mathcal{N}]$.
$P_G^{\max}$	Maximum generator output vector, $[P_{G_i}^{\max}, i \in \mathcal{N}]$.
P_D	Power load vector, $[P_{D_i}, i \in \mathcal{N}]$.
Θ	Voltage angle vector.
θ_i	Voltage angle for bus i.
$\mathbf{B}$	Admittance matrix.
x_{ij}	Line reactance from bus i to j.
$P_{Tij}^{\max}$	Line transmission limit from bus i to j.
N_{hid}	The number of hidden layers in the neural network.

We use $|\cdot|$ to denote the size of a set.

I. INTRODUCTION

Deep Neural Networks (DNNs) achieve superb performance in various complex engineering tasks [1]. Their capability of approximating any continuous mapping and their scalability make DNNs a favorable choice for predicting solutions to challenging large-scale optimization problems. Motivated by this, the learning-based approaches for the OPF problem were proposed and achieve desirable performances [2], [3]. However, to ensure the feasibility of the obtained solutions to system constraints is the main challenge. For example, the existing schemes can work well in light-load regimes (i.e., the system constraints are not highly binding). However, they may generate infeasible solutions due to the inevitable approximation errors, especially in high-load regimes (i.e., the system constraints are highly binding). It may lead to an undesirable increase in computational time, as inaccurate predictions require a high-computational post-processing procedure to restore feasibility. To address this issue, we propose a preventive learning approach named DeepOPF+. The advantage of the proposed approach is that ensuring the feasibility of the solution without relying on a computationally expensive post-processing procedure by calibrating the system constraints (including the generation and line limits) used in training. The proposed DeepOPF+ improves upon existing approaches [2] as it ensures feasibility in both light-load and heavy-load regimes while at the same time achieving significant computational speed-ups. The DeepOPF+ approach can apply to more general settings, including security-constrained OPF [3] and non-convex AC-OPF problems, which we leave for future studies.

We summarize our main contributions in the following. First, after reviewing the DC-OPF problem in Sec. III, we propose the preventive learning framework for DeepOPF+ in Sec. IV. Specifically, we introduce the preventive calibration of constraint limits during the training to ensure the feasibility of the predicted solutions. We remark that for each power network, we train a DNN model to approximate the corresponding load-generation mapping and predict the generations from the load inputs. Second, as described in Sec. IV-B, we provide a theoretical analysis of the relationship between the preventive constraint calibration and the approximation error of DNN. Finally, in Sec. V, simulation results using IEEE 30, 118, 200, and 300 bus test cases show that DeepOPF+ generates 100% feasible solutions with minor optimality loss under suitable calibration. Meanwhile, it achieves a *two orders of magnitude* computational speed-up compared to conventional approaches.

II. RELATED WORK

Machine learning, including neural networks, has been applied to challenging power system problems for decades, for a comprehensive review please refer to [4]. The recent advances made in deep learning have renewed interest in applications for power systems [5]. For brevity, we focus here on learning-based methods for solving OPF problems. The current learning-based work consists of two categories. The first category is a hybrid approach, which integrates the learning techniques into the conventional solution algorithm to solve challenging OPF problems [6]–[19]. However, the core of these methods is still the traditional solver, which may incur high computational costs for large-scale power systems.

The second category is the end-to-end approach, which leverages machine learning models to predict solutions to OPF problems directly [2], [3], [20]–[24]. The main challenge is to ensure that the predicted solutions satisfy the equality and inequality constraints. Existing works such as [2], [3] introduced a post-processing procedure to handle this issue, which, however, can still be computationally expensive. To the best of our knowledge, developing the end-to-end DNN approach to solving the DC/SCDC-OPF problems is first proposed in [2], [3], where a predict-and-reconstruct framework was proposed. The work in [21] applies the framework in [2], [3] to solve AC-OPF problems, but without considering the operating constraints on generations/line flows, which leads to a substantial fraction of infeasible solutions for the test cases. Recently, the authors in [25] generalize the predict-and-reconstruct framework to the AC-OPF settings and explores a penalty approach with zero-order optimization techniques to significantly improve the feasibility of the obtained AC-OPF solutions. The zero-order optimization techniques address the challenge of not having an explicit form of the penalty terms related to the generations/line constraint violations and thus not able to apply the conventional first-order techniques like stochastic gradient decent. Different from the existing end-to-end learning-based approaches, our proposed DeepOPF+ systematically calibrates constraint limits during the training stage. The proposed approach can ensure the feasibility of the obtained solutions without involving any post-processing procedure in the test stage. In our proposed DeepOPF+, we consider the line limits and the generation limits, add a loss term penalizing constraint violations and systematically calibrate constraint limits during training to obtain universal feasibility and consistent speedups. We also provide a theoretical analysis of the required constraint calibration.

III. The DC-OPF Problem

The DC-OPF problem [26] can be formulated as follows:

$$\min_{\mathbf{P_G},\Theta} \quad \sum_{i\in\mathcal{G}} g_i\left(P_{Gi}\right) \tag{1}$$

$$\text{s.t.} \quad P_{Gi}^{\min} \le P_{Gi} \le P_{Gi}^{\max},\ \forall i \in \mathcal{G}, \tag{2}$$

$$\mathbf{B}\cdot\Theta = \mathbf{P_G} - \mathbf{P_D}, \tag{3}$$

$$\frac{1}{x_{ij}}\left(\theta_i - \theta_j\right) \le P_{Tij}^{\max},\ \forall (i,j)\in\mathcal{E}. \tag{4}$$

The first set of constraints in the formulation describe the generation limits. The second set of constraints are the DC power flow equations. The third set of constraints capture the line transmission capacity. In the objective, $g_i\left(P_{Gi}\right)$ is the cost

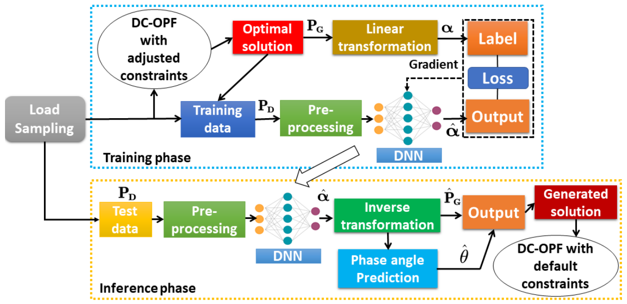

Fig. 1. The feasibility-ensuring learning framework of DeepOPF+. $\mathbf{P_D}$ is the sampled load vector. $\mathbf{P_G}$ ($\hat{\mathbf{P}}_\mathbf{G}$) is the optimal (predicted) bus power generation vector and α ($\hat{\alpha}$) is the corresponding optimal (predicted) scaling factors. $\hat{\theta}$ is the generated voltage phase angle of each bus.

function for the generator at the i-th bus, commonly modeled as a quadratic function [27]:

$$g_i\left(P_{Gi}\right) = \lambda_{1i}P_{Gi}^2 + \lambda_{2i}P_{Gi} + \lambda_{3i}, \quad \forall i \in \mathcal{G} \tag{5}$$

where λ_{1i}, λ_{2i}, and λ_{3i} are the model parameters and can be measured from data of the heat-rate curve [28]. Noted that the DC-OPF problem is a strictly convex (quadratic) problem and thus has a unique optimal solution. Numerical iteration solvers e.g., interior-point methods [29] can be applied to obtain the optimal solutions. Due to the increased uncertainty from renewable generation and stochastic loads, system operators have to solve OPF closer to real-time, posing computational challenges for conventional optimization solvers, motivating the use of learning-based approaches.

IV. DeepOPF+ for Solving DC-OPF

A. *Overview of DeepOPF+*

The methodology of the proposed DeepOPF+ can be divided into training and inference stages. We remark that in DeepOPF+, we first train a DNN to learn the load-generation mapping. The key novelty in DeepOPF+ is the introduction of the feasibility-ensuring learning framework as depicted in Fig. 1. We will describe the process and the analysis of DeepOPF+ in following subsections. In the inference stage, we directly apply DeepOPF+ to solve the DC-OPF problem with given test load inputs.

B. *Learning Framework for Ensuring Feasibility*

1) Constraints Calibration and Load Sampling: DeepOPF+ is based on a novel learning framework in which the system constraints are adjusted preventively during the training stage to ensure the feasibility of the predicted solutions during the test stage. The idea is that we first calibrate the system constraints, e.g., the transmission line and slack bus generator's output regarding capacity limits by an appropriate magnitude during the load sampling. As discussed in Sec. IV-D, the line capacity limits are reduced by a certain magnitude η_{ij}, i.e., $\frac{1}{x_{ij}}\left(\theta_i - \theta_j\right) \le P_{Tij}^{\max} - \eta_{ij}$, $\forall\left(i,j\right) \in \mathcal{E}$. The slack bus generation limits should also be calibrated with ξ, i.e., $P_{G_s}^{min} + \xi \le P_{G_s} \le P_{G_s}^{max} - \xi$, where P_{G_s} is the output of slack bus and $P_{G_s}^{min}$ and $P_{G_s}^{min}$ are the corresponding generation limits. Our theoretical analysis characterizes the necessary calibration magnitude for ensuring universal feasibility. Then, we train the DNN on a dataset created with calibrated limits and evaluate its performance on a test dataset with the original limits. Thus, even with the inherent prediction error of DNN, the obtained solution can still remain feasible.

2) Linear transformation and mapping dimension reduction: We first reformulate the inequality constraints on active generator power through linear scaling [2]:

$$P_{Gi}(\alpha_i) = \alpha_i \cdot \left(P_{Gi}^{\max} - P_{Gi}^{\min}\right) + P_{Gi}^{\min},\ \alpha_i \in [0,1], i \in \mathcal{G}, \tag{6}$$

Then, we leverage the fact that the admittance matrix (after removing the entries corresponding to the slack bus) is full rank to express the phase angles of all buses (except the phase angle of the slack bus) as following:

$$\tilde{\Theta} = \left(\tilde{\mathbf{B}}\right)^{-1}\left(\tilde{\mathbf{P}}_{\mathbf{G}} - \tilde{\mathbf{P}}_{\mathbf{D}}\right), \tag{7}$$

where $\tilde{\mathbf{P}}_{\mathbf{G}}$ and $\tilde{\mathbf{P}}_{\mathbf{D}}$ are the $(N-1)$-dimensional generation and load vectors for all buses except the slack bus. We output the N-dimensional phase angle vector Θ by inserting a zero phase angle for the slack bus into $\tilde{\Theta}$. Therefore, the voltage phase angles can be inferred directly from the predicted generator set-points. As benefits, the size of the DNN model and the amount of training data and time can be reduced.

3) The DNN model: The DNN model is based on a multi-layer feed-forward neural network structure, which consists of a typical three-level architecture: an input layer, several hidden layers, and an output layer. We use Rectified Linear Unit (ReLU) as non-linear activation functions of the neurons in the hidden layers. At the last step in the output layer, the Sigmoid function is applied to project the neural network output to the interval $(0, 1)$ for the scaling factor prediction.

After constructing the DNN model, we design the corresponding loss function used in the training. The loss function consists of two parts. The first part is the sum of mean square error between each element in the generated scaling factors $\hat{\alpha}_i$ and the actual scaling factors α_i of the optimal solutions:

$$\mathcal{L}_{PG} = \frac{1}{|\mathcal{G}|}\sum_{i\in\mathcal{G}}(\hat{\alpha}_i - \alpha_i)^2. \tag{8}$$

Meanwhile, we introduce a penalty term related to the inequality constraint into the loss function. We first introduce an $N_a \times N$ matrix $\mathbf{A}$ derived from the line admittance matrix [30], where N_a is the number of adjacent buses. Each row in $\mathbf{A}$ corresponds to an adjacent bus pair. Given the k-th adjacent bus pair $(i_k, j_k) \in \mathcal{E}$, $k = 1, ..., N_a$, let the power flow from the i_k-th bus to the j_k-th bus. Thus, the elements, a_{ki_k} and a_{kj_k}, the corresponding (k, i_k) and (k, j_k) entries of the matrix $\mathbf{A}_c$, are given as:

$$a_{ki_k} = \frac{1}{P_{Ti_kj_k}^{\max} \cdot x_{i_kj_k}} \text{ and } a_{kj_k} = \frac{-1}{P_{Ti_kj_k}^{\max} \cdot x_{i_kj_k}}. \tag{9}$$

Specifically, the elements of each row in the line admittance matrix are divided by the corresponding line capacity limit, respectively. Based on (7) and (9), the capacity constraints for the transmission lines in (4) can be expressed as:

$$-1 \le \left(\mathbf{A}\hat{\Theta}\right)_k \le 1, k = 1, ..., N_a, \tag{10}$$

where $\left(\mathbf{A}\hat{\Theta}\right)_k$ represents the k-th element of $\mathbf{A}\hat{\Theta}$. Note that $\hat{\Theta}$ is the phase angle vector generated based on (7), and it is computed from $\tilde{\mathbf{P}}_{\mathbf{G}}$ and $\tilde{\mathbf{P}}_{\mathbf{D}}$. We can then calculate $\left(\mathbf{A}\hat{\Theta}\right)_k$. The penalty term capturing the feasibility of the generated solutions can be expressed as:

$$\mathcal{L}_{pen} = \frac{1}{N_a}\sum_{k=1}^{N_a}\max\left(\left(\mathbf{A}_c\hat{\Theta}_c\right)_k^2 - 1, 0\right). \tag{11}$$

In summary, the loss function consists of two parts: the difference between the generated solution and the reference solution and the penalty upon solutions violating the inequality constraints. The total loss is a weighted sum of the two:

$$\mathcal{L}_{total} = w_1 \cdot \mathcal{L}_{PG} + w_2 \cdot \mathcal{L}_{pen}, \tag{12}$$

where w_1 and w_2 are positive weighting factors. 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, which include each layer's connection weight matrix and bias vector. We apply the widely-used stochastic gradient descent (SGD) method with momentum [31] method to update the DNN's parameters at each iteration. We refer to [3] for details of the DNN structure and the training process of optimizing DNN's parameters.

C. Computational Complexity

Recall that N is the number of buses. The number of optimization variables in DCOPF, including the generations and the phase angles of all buses is $\mathcal{O}(N)$. The computational complexity of interior point methods for solving DCOPF as a convex quadratic problem is $\mathcal{O}\left(N^4\right)$, measured as the number of elementary operations assuming that each elementary operation takes a fixed amount of time to perform [29].

The computational complexity of DeepOPF+ consists of three parts. The first is the complexity of predicting the generations using the DNN, which is $\mathcal{O}\left(N_{hid}M^2\right)$ where M is the maximum number of neurons in each layer and N_{hid} is the number of hidden layers in DNN. See [3] for details of the analysis. To achieve satisfactory performance in terms of optimality loss and speed-up, we set M to be $\mathcal{O}(N)$ and N_{hid} to be 3. As such, the complexity for predicting the generations by our DNN is $\mathcal{O}\left(N^2\right)$.

The second is the complexity of computing the phase angles from the generations by directly solving (linearized) power flow equations and checking the feasibility of the results. The total complexity is $\mathcal{O}\left(N^3\right)$.

The third is the complexity of ℓ_1-projection, if the post-processing procedure is involved to ensure feasibility of the obtained solutions. Please refer to [3] for details of ℓ_1-projection process. We note that under a proper constraint limits calibration magnitude, DeepOPF+ can always provide feasible solutions without post-processing as shown in Sec. V. The ℓ_1-projection is a linear programming problem and can be solved in $\mathcal{O}\left(N^{2.5}\right)$ amount of time by using algorithms based on fast matrix multiplication [32].

Overall, the total computational complexity of DeepOPF+ is $\mathcal{O}\left(N^3\right)$. which is lower than that of the conventional interior point method, which is $\mathcal{O}\left(N^4\right)$.

D. Theoretical analysis

In this section, we provide a theoretical analysis on the error transfer between the prediction errors of the generator set-points obtained from DNN and the power flow mismatch on each transmission line and the slack bus generator output. In addition, we show that given exact bounds of the prediction errors of the generator set-points, the maximal power offsets among all lines and the slack bus generation can be obtained (i.e., the required magnitude by which the line capacity and slack bus generation range have to be reduced in the training stage to ensure feasibility in the test stage). To quantify the relationship between prediction errors and power offset on lines and slack bus, we provide the following theorem:

Theorem 1. *Let ϵ be the maximum prediction error of the DNN such that $|\hat{P}_{Gi} - P_{Gi}| \le \epsilon$ holds for all $i = 1, 2, ..., N$, where $\hat{P}_{Gi}$ is the predicted generators' output. We have*

- *the maximum power offsets on the i-th line are $k_i \cdot \epsilon$, where $k_i = \sum_{k=1}^{N-1} |M_{ik}|$, $i \in \mathcal{E}$. Thus, if the calibration for the i-th line capacity constraint is set to be no less than $k_i \cdot \epsilon$, $\hat{P}_G$ satisfies the line capacity constraints*
- *the maximum power offset on the slack bus generation is $(|\mathcal{G}| - 1) \cdot \epsilon$, where $|\mathcal{G}|$ denotes the number of generators. Thus, if the calibration for the slack bus generation constraint is set to be no less than $(|\mathcal{G}|-1)\cdot\epsilon$, $\hat{P}_G$ satisfies the slack bus generation constraints.*

The matrix M is $|\mathcal{E}| \times (N-1)$ dimensional depending on the topology of the power network with entries of Power Transfer Distribution Factors (PTDFs), and $|\mathcal{E}|$ is the number of transmission lines. Please refer to the technical report [33] for a complete proof and detailed formula of M. For example, the maximum $\max_{i\in\mathcal{E}} k_i = 19, 52$, 199, and 299 for IEEE 30-, 118-, 200-, and the 300-cases used in our simulation, respectively. The maximum prediction error of the DNN depends on the neural network architecture as defined in the following theorem.

Theorem 2. *Reproduced from [3]: Let $\mathcal{H}$ be the class of all possible $f^*(\cdot)$ with a Lipschitz constant $\Lambda > 0$. Let $\mathcal{K}$ be the class of all $f(\cdot)$, generated by a neural network with depth N_{hid} and maximum number of neurons per layer N_n. Then, the maximum prediction error can be defined as:*

$$\epsilon = \max_{f^*\in\mathcal{H}} \min_{f\in\mathcal{K}} \max_{x\in\mathcal{S}} |f^*(x) - f(x)| \ge \Lambda \cdot \frac{d}{4\cdot(2N_n)^{N_{hid}}}, \quad (13)$$

where d is the diameter of the load input domain $\mathcal{S}$.

The theorem characterizes a lower bound on the worst-case error of using neural networks to approximate load-to-generation mappings in DC-OPF problems. Such prediction error ϵ is characterized by the DNN structure and hence can be pre-obtained before training. Based on Theorem 1 and 2, we derive the following lemma.

TABLE I
PARAMETERS FOR TEST CASES.

Case	N	$\lvert\mathcal{G}\rvert$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{K}\rvert$	N_{hid}	Neurons per hidden layer
Case30	30	6	2	41	3	32/16/8
Case118	118	19	99	186	3	128/64/32
Case200	200	32	108	245	3	128/64/32
Case300	300	57	199	411	3	256/128/64

* The number of load buses is calculated based on the default load on each bus. A bus is considered a load bus if its default active power consumption is non-zero.

Lemma 1. *To guarantee the feasibility of the DNN approximating the most difficult load-to-generation mapping with a Lipschitz constant Λ,*

- *the capacity constraint for the i-th line is required to be reduced during the training stage at least by* $k_i \cdot \epsilon = k_i \cdot \Lambda \cdot \frac{d}{4\cdot(2N_n)^{N_{hid}}}$, $i \in \mathcal{E}$,
- *the slack bus generation limits are required to be calibrated during the training stage at least by* $(|\mathcal{G}| - 1) \cdot \epsilon = (|\mathcal{G}| - 1) \cdot \Lambda \cdot \frac{d}{4\cdot(2N_n)^{N_{hid}}}$,

where P_{G_s} is the output of slack bus and $P_{G_s}^{min}$ and $P_{G_s}^{min}$ are the corresponding generation limits.

The proof of Lemma 1 follows directly from Theorem 2 and Theorem 1. It indicates that the line capacity limits and the slack bus generation limits are required to be calibrated by a necessary magnitude such that even with prediction errors, DNNs could still generate feasible solutions under the worst-case. Lemma 1 provides the insight that larger NN sizes contribute to smaller prediction errors and thus require smaller magnitudes of reduction of system constraints to ensure universal feasibility. Note that in practice, it is challenging to determine tight worst-case prediction errors of DNNs. We plan to compute the Lipschitz constants of DNNs [34] and tight bounds on the approximation error and exact constraints calibration magnitudes in future work. In the next section, we show numerical results for five illustrative transmission line capacity limits and slack bus generation limits calibrations without prior knowledge of approximation error ϵ.

V. NUMERICAL EXPERIMENTS

A. Experiment setup

1) Simulation environment: The experiments are conducted in CentOS 7.6 on the quad-core (i7-3770@3.40G Hz) CPU workstation with 16GB RAM.

2) Test case: The proposed approach is evaluated for four representative test cases: IEEE 30-bus, 118-bus test cases [35], a 200-bus power system [36], and IEEE 300-bus test case.[1]

[1]As IEEE 118-bus and 300-bus test cases provided by MATPOWER [37] do not specify the line capacities, we use IEEE 118-bus test case provided by Power Grid Lib [38] and use the line capacity setting for IEEE 300-bus test case with same branch from Power Grid Lib [38] (version 19.05).

TABLE II
PERFORMANCE EVALUATION OF DEEPOPF+.

Case	Limit calibration (%)	Feasibility rate (%)	Feasibility rate without calibration (%)	Average cost ($/hr)			Average running time (ms)		Average speedup
				DeepOPF+	Ref.	Loss(%)	DeepOPF+	Ref.	
	0.5	94.94		679.0		0.27	0.54		$\times 85$
	1.5	96.90		679.0		0.27	0.52		$\times 86$
Case30	3.5	100	94.02	679.0	677.3	0.27	0.50	44	$\times 88$
	5	100		679.0		0.27	0.49		$\times 88$
	7	100		679.2		0.30	0.50		$\times 89$
	0.5	70.08		111617.6		0.36	1.60		$\times 143$
	1.5	81.12		111660.8		0.40	1.27		$\times 152$
Case118	3.5	97.72	61.66	111752.7	111219.5	0.48	0.62	116	$\times 205$
	5	100		111829.6		0.55	0.59		$\times 200$
	7	100		111925.6		0.63	0.56		$\times 208$
	0.5	68.58		39118.0		0.95	2.09		$\times 114$
	1.5	86.52		39257.3		1.31	1.18		$\times 158$
Case200	3.5	91.2	63.36	39319.0	38754.7	1.47	0.96	104	$\times 167$
	5	94.62		39388.6		1.65	0.81		$\times 175$
	7	100		39501.3		1.94	0.59		$\times 178$
	0.5	78.14		853607.1		0.11	3.35		$\times 95$
	1.5	86.96		583581.6		0.11	2.23		$\times 111$
Case300	3.5	97.92	75.94	853831.8	852611.6	0.14	0.96	82	$\times 118$
	5	100		854187.1		0.19	0.66		$\times 125$
	7	100		854998.5		0.28	0.66		$\times 126$

* If the DNN generates infeasible solutions, we apply an efficient ℓ_1-projection post-processing procedure to ensure the feasibility of the final solution [3] by the Gurobi solver. The average running time includes the post-processing time if DNN obtains infeasible solutions.

* Speedup is calculated as the average of the running-time ratios of Pypower to DeepOPF+ for all the test instances. We note that the speedup is the average of ratios, and it is different from the ratio of the average running times between Pypower and DeepOPF+.

* Note that Case118 takes longer computational time to obtain the optimal solution with the conventional solver compared to Case300. This is due to the observation that Case118 requires more iteration steps to converge (on average 25 times) than Case300 (on average 11 times), while the average running time per iteration of Case118 (4.7 ms) is less than that of Case300 (7.5 ms).

3) Training data: For the training stage, the load data is sampled within $[100\%, 130\%]$ of the default load on each load uniformly at random, which covers both light-load and heavy-load regimes. On the heavy-load regimes, some transmission lines and the slack bus generation will reach their upper operation limits under the given load. Based on preliminary experiments, we test DeepOPF+ by calibrating the transmission lines' limits and slack bus generation limits by $c = 0.5\%$, 1.5%, 3.5%, 5%, and 7%, respectively. Note that we will investigate the systematic calibration of the limits using Lemma 1 in future work. The solution for the DC-OPF problem provided by Pypower [39] is regarded as ground-truth. For each test case, the amount of training data and test data are 25'000 and 5'000.

4) The implementation of the DNN model: We design the DNN model based on the Pytorch platform. For the training process, the number of epochs is 200, and the batch size is 64. Based on the range of each loss obtained from preliminary experiments, the value of weighting factors w_1 and w_2 are each set to 1. For each power network, we train a DNN model to approximate the corresponding load-generation mapping. We remark that the DNN inputs the load profile and outputs the generation prediction. According to the size of the power network, we design DNN models with a different number of neural network layers. These parameters are given in Table I.

B. Performance evaluation

We show the simulation results in Table II. For the test cases, we can observe infeasibility if constraint calibrations are not applied during the training stage. With the preventive calibrations, the percentage of feasible solution is improvement, which is up to 38% (i.e., from 62% to 100%). It indicates the effectiveness of the proposed DeepOPF+ in ensuring the feasibility of the predicted solutions to the system constraints even with inevitable prediction errors. Also, the differences between the cost of the predicted solutions and that of the reference solutions is minor (at most 1.94%). Compared with the traditional DC-OPF solver, our DeepOPF+ approach reduces the computational time by two orders of magnitude. Note that the existing DNN-based schemes may not achieve high computational speedups for both light- and high-loading regime. They may predict infeasible solutions due to violating the operating constraints regarding, e.g., transmission line, and need to resort to post-processing to recover feasible solutions. We show the results of the comparisons for light-load and heavy-load regimes in the technical report [33].

Moreover, we observe that a larger calibration magnitude contributes to a higher feasibility rate but larger optimality loss. It could be interpreted that after constraint calibrations, the DNN approximates the mapping from load inputs to sub-optimal solutions of the adjusted DC-OPF problems. It indicates the trade-off between ensuring the feasibility and maintaining minor optimality loss of the predicted solutions. We remark the importance of determining the minimal calibration magnitude such that the DeepOPF+ scheme can achieve satisfactory speedup performance with minor optimality loss. In our test cases, we found that different cases have different minimal calibration magnitudes. For example, DeepOPF+ achieves 100% feasibility rate for Case30 with a 3.5% calibration magnitude. For Case118 and Case300, a 5% calibration magnitude guarantees the 100% feasibility rates of DeepOPF+. For Case200, with a 7% calibration magnitude, the predicted solutions of DeepOPF+ are all feasible. Under our test case setting, if the constraints calibration magnitude is 7%, the feasibility percentages of the predicted solutions by DeepOPF+ are 100% over the four test cases.

VI. Conclusion

In this paper, we propose DeepOPF+ as a preventive learning approach for solving DC-OPF problems. Our main contributions are that we ensure the feasibility of the predicted solutions by systematically calibrating the constraint limits in the training stage. We theoretically characterize a necessary condition for the magnitude of reduction to guarantee feasibility. Simulation results show that DeepOPF+ achieves a computational speed-up by two orders of magnitude as compared to conventional solvers with minor optimality loss and ensures feasibility for physical system constraints. Extending the DeepOPF+ scheme to solve the general AC-OPF problems is an immediate future direction.

References

[1] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, Deep Learning. MIT Press Cambridge, 2016, vol. 1.

[2] X. Pan, T. Zhao, and M. Chen, "DeepOPF: Deep Neural Network for DC Optimal Power Flow," in 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm). IEEE, 2019, pp. 1–6.

[3] ——, "Deepopf: A Deep Neural Network Approach for Security-Constrained DC Optimal Power Flow," arXiv preprint arXiv:1910.14448, 2019.

[4] L. A. Wehenkel, Automatic learning techniques in power systems. Springer Science & Business Media, 2012.

[5] F. Li and Y. Du, "From alphago to power system ai: What engineers can learn from solving the most complex board game," IEEE Power and Energy Magazine, vol. 16, no. 2, pp. 76–84, 2018.

[6] V. J. Gutierrez-Martinez, C. A. Cañizares, C. R. Fuerte-Esquivel, A. Pizano-Martinez, and X. Gu, "Neural-network security-boundary constrained optimal power flow," IEEE Transactions on Power Systems, vol. 26, no. 1, pp. 63–72, 2010.

[7] A. Vaccaro and C. A. Cañizares, "A knowledge-based framework for power flow and optimal power flow analyses," IEEE Transactions on Smart Grid, vol. 9, no. 1, pp. 230–239, 2016.

[8] L. Halilbašić, F. Thams, A. Venzke, S. Chatzivasileiadis, and P. Pinson, "Data-driven security-constrained ac-opf for operations and markets," in 2018 Power Systems Computation Conference (PSCC). IEEE, 2018, pp. 1–7.

[9] D. Biagioni, P. Graf, X. Zhang, and J. King, "Learning-Accelerated ADMM for Distributed Optimal Power Flow," arXiv preprint arXiv:1911.03019, 2019.

[10] K. Baker, "Learning warm-start points for ac optimal power flow," arXiv preprint arXiv:1905.08860, 2019.

[11] M. Jamei, L. Mones, A. Robson, L. White, J. Requeima, and C. Ududec, "Meta-optimization of optimal power flow," ICML Workshop, Climate Change: How Can AI Help, 2019.

[12] D. Deka and S. Misra, "Learning for dc-opf: Classifying active sets using neural nets," arXiv preprint arXiv:1902.05607, 2019.

[13] S. Karagiannopoulos, P. Aristidou, and G. Hug, "Data-driven local control design for active distribution grids using off-line optimal power flow and machine learning techniques," IEEE Transactions on Smart Grid, vol. 10, no. 6, pp. 6461–6471, 2019.

[14] K. Baker and A. Bernstein, "Joint chance constraints in ac optimal power flow: Improving bounds through learning," IEEE Transactions on Smart Grid, vol. 10, no. 6, pp. 6376–6385, 2019.

[15] Y. Ng, S. Misra, L. A. Roald, and S. Backhaus, "Statistical learning for DC optimal power flow," in 2018 Power Systems Computation Conference (PSCC). IEEE, 2018, pp. 1–7.

[16] S. Misra, L. Roald, and Y. Ng, "Learning for Constrained Optimization: Identifying Optimal Active Constraint Sets," arXiv preprint arXiv:1802.09639, 2018.

[17] Q. Zhai, X. Guan, J. Cheng, and H. Wu, "Fast identification of inactive security constraints in SCUC problems," IEEE Transactions on Power Systems, vol. 25, no. 4, pp. 1946–1954, 2010.

[18] L. A. Roald and D. K. Molzahn, "Implied constraint satisfaction in power system optimization: the impacts of load variations," in 2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton). IEEE, 2019, pp. 308–315.

[19] S. Pineda, J. Morales, and A. Jiménez-Cordero, "Data-Driven Screening of Network Constraints," arXiv preprint arXiv:1907.04694, 2019.

[20] N. Guha, Z. Wang, M. Wytock, and A. Majumdar, "Machine learning for AC optimal power flow," arXiv preprint arXiv:1910.08842, 2019.

[21] A. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," arXiv preprint arXiv:1910.01213, 2019.

[22] F. Fioretto, T. W. Mak, and P. Van Hentenryck, "Predicting AC Optimal Power Flows: Combining Deep Learning and Lagrangian Dual Methods," arXiv preprint arXiv:1909.10461, 2019.

[23] R. Dobbe, O. Sondermeijer, D. Fridovich-Keil, D. Arnold, D. Callaway, and C. Tomlin, "Toward distributed energy services: Decentralizing optimal power flow with machine learning," IEEE Transactions on Smart Grid, vol. 11, no. 2, pp. 1296–1306, 2019.

[24] E. R. Sanseverino, M. Di Silvestre, L. Mineo, S. Favuzza, N. Nguyen, and Q. Tran, "A multi-agent system reinforcement learning based optimal power flow for islanded microgrids," in 2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC). IEEE, 2016, pp. 1–6.

[25] X. Pan, M. Chen, T. Zhao, and S. H. Low, "DeepOPF: A Feasibility-Optimized Deep Neural Network Approach for AC Optimal Power Flow Problems," arXiv preprint arXiv:2007.01002, 2020.

[26] A. Gomez-Exposito, A. J. Conejo, and C. Canizares, Electric Energy Systems: Analysis and Operation. CRC press, 2018.

[27] 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.

[28] 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.

[29] 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.

[30] B. Stott, J. Jardim, and O. Alsac, "Dc power flow revisited," IEEE Transactions on Power Systems, vol. 24, no. 3, pp. 1290–1300, Aug 2009.

[31] N. Qian, "On the momentum term in gradient descent learning algorithms," Neural networks, vol. 12, no. 1, pp. 145–151, 1999.

[32] P. M. Vaidya, "Speeding-up linear programming using fast matrix multiplication," in IEEE FOCS, 1989, pp. 332–337.

[33] T. Zhao, X. Pan, M. Chen, A. Venzke, and S. H. Low, "DeepOPF+: A Deep Neural Network Approach for DC Optimal Power Flow for Ensuring Feasibility," arXiv preprint arXiv:2009.03147, 2020.

[34] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas, "Efficient and accurate estimation of lipschitz constants for deep neural networks," in Advances in Neural Information Processing Systems, 2019, pp. 11 423–11 434.

[35] "Power Systems Test Case Archive," 2018, http://labs.ece.uw.edu/pstca/.

[36] A. B. Birchfield, T. Xu, K. M. Gegner, K. S. Shetye, and T. J. Overbye, "Grid Structural Characteristics as Validation Criteria for Synthetic Networks," IEEE Transactions on Power Systems, vol. 32, no. 4, pp. 3258–3265, 2017.

[37] R. D. Zimmerman, C. E. Murillo-Sánchez, R. J. Thomas et al., "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," IEEE Transactions on Power Systems, vol. 26, no. 1, pp. 12–19, 2011.

[38] S. Babaeinejadsarookolaee, A. Birchfield, R. D. Christie, C. Coffrin, C. DeMarco, R. Diao, M. Ferris, S. Fliscounakis, S. Greene, R. Huang et al., "The power grid library for benchmarking ac optimal power flow algorithms," arXiv preprint arXiv:1908.02788, 2019.

[39] "pypower," 2018, https://pypi.org/project/PYPOWER/.

APPENDIX A
PROOF OF THEOREM 1

Proof. We prove the two claims one by one. For the first claim, the power flow from i-th bus to j-th bus is $\frac{1}{x_{ij}}(\theta_i - \theta_j)$. Given the predicted generation profile, the phase angles can be recovered by the linear relationship (7). Therefore, the power flows on the power network lines are given as

$$PF = X \left[\begin{array}{c} 0 \\ \left(\tilde{\mathbf{B}}\right)^{-1}\left(\tilde{\mathbf{P}}_{\mathbf{G}} - \tilde{\mathbf{P}}_{\mathbf{D}}\right) \end{array} \right] = \tilde{X}\left(\tilde{\mathbf{B}}\right)^{-1}\left(\tilde{\mathbf{P}}_{\mathbf{G}} - \tilde{\mathbf{P}}_{\mathbf{D}}\right),$$

where X is a $|\mathcal{E}| \times N$ matrix and $X_{ai} = x_{ij}, X_{aj} = -x_{ij}$ if there exist a line between i-th bus to the j-th bus and otherwise 0. $\tilde{X}$ is the matrix eliminating the column corresponds to the slack bus of matrix X. Therefore, the power offsets on each line due to the prediction error of P_G can be expressed as:

$$|\hat{PF} - PF| = \left|\tilde{X}\left(\tilde{\mathbf{B}}\right)^{-1}(\hat{P}_{\mathbf{G}} - \tilde{P}_{\mathbf{G}})\right|. \tag{14}$$

For simplicity, we use M to denote $\tilde{X}\left(\tilde{\mathbf{B}}\right)^{-1}$. Given the maximal prediction error ϵ, the maximal power offset on the i-th line is given as

$$\sum_{k=1}^{N-1} |M_{ik}| \cdot \epsilon = k_i \cdot \epsilon,$$

where $k_i = \sum_{k=1}^{N-1} |M_{ik}|$.

For the second claim, the slack bus generation is given as

$$P_{G_s} = \sum_{i\in\mathcal{D}} P_{D_i} - \sum_{j\in\mathcal{G}, j\neq s} P_{G_j}. \tag{15}$$

Given the maximal prediction error ϵ, the maximal power offset on slack bus is given as

$$\max\left|\hat{P_{G_s}} - P_{G_s}\right| = \max\left|\sum_{j\in\mathcal{G},j\neq s}(\hat{P_{G_j}} - P_{G_j})\right| = (|\mathcal{G}-1|)\cdot\epsilon,$$

where s denotes the slack bus index. This completes the proof. □

APPENDIX B
THE SPEEDUP COMPARISON FOR LIGHT-LOAD AND HEAVY-LOAD REGIMES

We also carry out comparative experiments to show the benefits brought by the calibration of constraint limits in the training. More specifically, we compare the speedups performance of the following six schemes:

- DeepOPF: The previous DNN approach for DC-OPF problem, which uses the default limits in the training and a post-processing procedure to ensure the feasibility of the obtained solutions.
- DeepOPF+_$V1$, DeepOPF+_$V2$, DeepOPF+_$V3$, DeepOPF+_$V4$, DeepOPF+_$V5$: The proposed DeepOPF+ approach with 0.5%, 1.5%, 3.5%, 5%, and 7% line capacity limits and slack bus generation limits calibrations in the training stage, respectively.

Comparative experiments follow the same experimental settings above and the average speedups of the test instances are shown in Table III. We observe that DeepOPF+ improves over existing DNN-based schemes in that it achieves consistent speedups in both light-load and heavy-load regimes, in which the load data is sampled within $[90\%, 110\%]$ and $[110\%, 130\%]$, respectively. As mentioned before, existing DNN-based schemes may need a highly computational complexity post-processing procedure to ensure the feasibility in both light- and heavy-load regimes, which is not necessary in the proposed DeepOPF+.

TABLE III
SPEEDUPS PERFORMANCE EVALUATION OF DEEPOPF+ IN THE LIGHT- AND HEAVY- LOAD REGIMES.

Case	Scheme	Average Speedups	
		light-load regime	heavy-load regime
Case30	DeepOPF	×85	×83
	DeepOPF+_$V1$	×85	×83
	DeepOPF+_$V2$	×85	×85
	DeepOPF+_$V3$	×85	×89
	DeepOPF+_$V4$	×85	×89
	DeepOPF+_$V5$	×86	×89
Case118	DeepOPF	×104	×155
	DeepOPF+_$V1$	×103	×191
	DeepOPF+_$V2$	×103	×247
	DeepOPF+_$V3$	×104	×252
	DeepOPF+_$V4$	×103	×251
	DeepOPF+_$V5$	×166	×254
Case200	DeepOPF	×57	×105
	DeepOPF+_$V1$	×60	×109
	DeepOPF+_$V2$	×67	×138
	DeepOPF+_$V3$	×114	×177
	DeepOPF+_$V4$	×135	×208
	DeepOPF+_$V5$	×141	×208
Case300	DeepOPF	×65	×70
	DeepOPF+_$V1$	×79	×84
	DeepOPF+_$V2$	×131	×94
	DeepOPF+_$V3$	×143	×112
	DeepOPF+_$V4$	×148	×132
	DeepOPF+_$V5$	×145	×140