Learning to Solve the AC-OPF using Sensitivity-Informed Deep Neural Networks

Manish K. Singh *Member, IEEE*, Vassilis Kekatos *Senior Member, IEEE*, and Georgios B. Giannakis *Fellow, IEEE*

Abstract—To shift the computational burden from real-time to offline in delay-critical power systems applications, recent works entertain the idea of using a deep neural network (DNN) to predict the solutions of the AC optimal power flow (AC-OPF) once presented load demands. As network topologies may change, training this DNN in a sample-efficient manner becomes a necessity. To improve data efficiency, this work utilizes the fact OPF data are not simple training labels, but constitute the solutions of a parametric optimization problem. We thus advocate training a sensitivity-informed DNN (SI-DNN) to match not only the OPF optimizers, but also their partial derivatives with respect to the OPF parameters (loads). It is shown that the required Jacobian matrices do exist under mild conditions, and can be readily computed from the related primal/dual solutions. The proposed SI-DNN is compatible with a broad range of OPF solvers, including a non-convex quadratically constrained quadratic program (QCQP), its semidefinite program (SDP) relaxation, and MATPOWER; while SI-DNN can be seamlessly integrated in other learning-to-OPF schemes. Numerical tests on three benchmark power systems corroborate the advanced generalization and constraint satisfaction capabilities for the OPF solutions predicted by an SI-DNN over a conventionally trained DNN, especially in low-data setups.

Index Terms—Sensitivity analysis; data efficiency; optimality conditions; non-linear OPF solvers.

I. INTRODUCTION

Power system operation involves routinely solving various renditions of the optimal power flow (OPF) task. Physical expansion of power networks, increasing number of dispatchable resources, and renewable generation-induced volatility call for solving large-scale OPF problems frequently. These problems naturally involve nonlinear alternating-current (AC) power flow equations as constraints. Such formulations, referred to as AC-OPF, are non-convex and are often computationally too expensive for real-time applications. Approximate formulations such as the linearized or so termed DC-OPF serve as the pragmatic resort for such setups.

A concerted effort towards handling AC-OPF efficiently has yielded popular nonlinear solvers such as MATPOWER [1], and efficient conic relaxations with optimality guarantees for frequently encountered problem instances [2], [3], [4], [5]. Despite significant advancements in numerical solvers, scalability of the AC-OPF may still be a challenge, particularly in online, combinatorial, and stochastic settings [6], [7]. To alleviate these issues, there has been growing interest in developing machine learning-based approaches (particularly deep learning) for OPF [6], [7], [8], [9], [10], [11], [12]; see [13] for recent applications. The primary advantage of machine learning-based approaches lies in the speed-up during the inference phase. For instance, compared to conventional solvers, deep learning-based approaches have offered speed-ups by factors as high as 200 for DC-OPF, and 35 for AC-OPF [9], [10].

To entirely bypass numerical solvers for the OPF, one can opt for unsupervised learning approaches, such as the ones suggested in [14], [15], [16], [17], [18]. However, the performance of OPF solvers for offline data generation and the availability of historical data with power utilities adequately motivate supervised learning. Nevertheless, there are two main challenges central to learning for OPF. First, traditional deep learning approaches are not amenable to enforcing constraints even for the training set. Predictions for OPF minimizers may have limited standalone value if the related constraints are violated. Second, power systems undergo frequent topological and operational changes as generating units can be (de)-committed and transmission lines or bus/line reactors can be switched. Such changes may require retraining a DNN potentially at short intervals [19]. Deep learning-based approaches are traditionally data-intense and frequent retraining for large systems may be prohibitive.

To cope with the first challenge, a DNN may be engaged to better initialize existing numerical solvers [12], or to predict active constraints and thus result in an OPF model with much reduced number of constraints [11], [20], [21]. Another group of approaches targets constraint satisfaction by penalizing constraint violations and the related Karush–Kuhn–Tucker (KKT) conditions [8], [10], or explicitly resorting to a Lagrangian dual scheme for DNN training [7], [15], [22]. The third alternative involves post-processing DNN predictions by projecting them using a power flow solver [8], [12]. Although the projected point satisfies the power flow equations, it may still violate engineering limits.

To cater to the second challenge of frequent model changes, sample-efficient learning models that generalize well are well

motivated. One way to achieve that is via meta-learning, according to which training datasets generated from diverse grid topologies are used to train a vector of meta-weights [19]. When it later comes to training a DNN to handle a particular topology, its weights are initialized to the values of meta-weights. Alternatively, the sample efficiency of learning models could be improved by prudently designing the DNN architecture upon leveraging prior information. For example, by seeking an input-convex DNN when the underlying input-output mapping is convex [23], or using DNNs that *unroll* an iterative optimization algorithm [24], or adopting graph-based priors [14], [25]. The previously mentioned approaches that design objective functions based on OPF constraints and KKT conditions may also be seen as including prior knowledge in training, thus enhancing learnability.

Recent works in the general area of *physics-informed learning* aim at incorporating prior information on the underlying data, not occurring in conventional training datasets [26]. Specifically, for learning solutions of differential equations, the derivatives of DNN output with respect to input carry obvious virtue. Suitably utilizing these derivatives yields significant advantages in such applications [26], [27]. While for differential equations derivatives with respect to time and/or space dimensions emerge naturally, sensitivities in an optimization problem have been underappreciated. To this end, our recent work proposed a novel approach of training sensitivity informed DNNs (SI-DNN), intended to learn OPF solutions [28]. Training SI-DNNs requires computing the sensitivities of OPF minimizers with respect to the input parameters. For DC-OPF posed as linear or quadratic programs, computing these sensitivities is simpler, and using these to train DNNs yielded remarkable improvements; see [28].

Contributions: The contributions of this work are on four fronts: *c1)* We put forth a novel approach for training DNNs to predict AC-OPF solutions by matching not only the OPF minimizers, but also their sensitivities (partial derivatives collected in a Jacobian matrix) with respect to the OPF problem parameters (e.g., load demands). *c2)* We compute the desired sensitivities for general nonlinear OPF formulations building upon classical works on perturbation analysis of continuous optimization problems [29], [30], [31]. Existing sensitivity results assume certain constraint qualifications that may not be satisfied by OPF instances. *c3)* We relax such assumptions and establish that the sensitivities of primal OPF solutions do exist under milder conditions. *c4)* In pursuit of globally optimal OPF solutions for training a DNN, we also study the SDP formulation of the AC-OPF and compute its sensitivities by utilizing the sensitivities of the related QCQP. Such shortcut obviates the difficulty of differentiating a conic program. It is worth stressing that the proposed sensitivity-informed methodology can be used in tandem with other learning approaches, such as those reviewed earlier imposing physics-inspired DNN architectures, penalizing constraint violations, or adopting meta-learning.

Motivation: The advantage of using a DNN to predict OPF minimizers is its computational speed. Running a DNN is significantly faster than running an OPF solver. Alternatively, the DNN prediction can be used to *warm-start* an OPF solver

and thus expedite its convergence. This speed-up advantage during testing (inference) is featured by any DNN. Sensitivity-informed DNNs are advocated here as a means to expedite training as well. This is possible because an SI-DNN can attain the same prediction accuracy as a plain DNN using much fewer samples. Our numerical tests demonstrate that depending on the network an SI-DNN requires 1/4 or even 1/10 of the training samples compared to a plain DNN. Such speed-up is significant in setups where there is not sufficient time for generating samples offline. Such setups could arise when a DNN is used to predict the setpoints of inverter-interfaced distributed energy resources and the feeder is reconfigured frequently; or when unit commitment decisions change regularly the set of online generators in a transmission system; or when the DNN is part of a stochastic unit commitment formulation and has to be retrained for the various commitment configurations visited by a branch-and-bound algorithm.

The rest of the paper is organized as follows: Section II presents the key methodology for SI-DNNs. Section III poses the AC-OPF as a parametric optimization taking the form of a non-convex QCQP. Section IV establishes that the sought sensitivies of the AC-OPF exist under mild conditions and explains how they can be readily computed. Section V computes the sensitivies of globally optimal AC-OPF solutions obtained via the SDP relaxation of the OPF. The numerical tests of Section VI contrast SI-DNN to conventionally trained DNNs using datasets generated by MATPOWER and the SDP-based OPF solver on three benchmark power systems. Conclusions are drawn in Section VII.

Notation: lower- (upper-) case boldface letters denote column vectors (matrices). Calligraphic symbols are reserved for sets. Symbol $^\top$ stands for transposition, vectors $\mathbf{0}$ and $\mathbf{1}$ are the all-zeros and all-ones vectors or matrices, and $\mathbf{e}_n$ is the n-th canonical vector of appropriate dimensions implied by the context. Operator $\mathcal{D}(\mathbf{x})$ returns a diagonal matrix with the entries of vector $\mathbf{x}$ on its main diagonal.

II. SENSITIVITY-INFORMED TRAINING FOR DNNs

Consider the task of optimally dispatching generators and flexible loads to meet power balance constraints $\mathbf{h}(\cdot)$ while enforcing engineering limits captured by $\mathbf{g}(\cdot)$. OPF boils down to a *parametric optimization problem* that has to be solved routinely for different values of the problem parameters being the time-varying renewable generation and loads. The problem can be abstracted as: Given a vector $\boldsymbol{\theta} \in \mathbb{R}^P$ of problem parameters or inputs, find an optimal dispatch $\mathbf{x}_{\boldsymbol{\theta}} \in \mathbb{R}^N$ consisting of voltages and generator setpoints as the minimizer

$$\mathbf{x}_{\boldsymbol{\theta}} \in \arg\min_{\mathbf{x}} \ f(\mathbf{x}; \boldsymbol{\theta}) \qquad (P_{\boldsymbol{\theta}})$$
$$\text{s.to} \ \ \mathbf{h}(\mathbf{x}; \boldsymbol{\theta}) = \mathbf{0} : \ \boldsymbol{\lambda}_{\boldsymbol{\theta}}$$
$$\mathbf{g}(\mathbf{x}; \boldsymbol{\theta}) \leq \mathbf{0} : \ \boldsymbol{\mu}_{\boldsymbol{\theta}}$$

where functions $f(\mathbf{x}; \boldsymbol{\theta})$, $\mathbf{h}(\mathbf{x}; \boldsymbol{\theta})$, and $\mathbf{g}(\mathbf{x}; \boldsymbol{\theta})$ are continuously differentiable with respect to $\mathbf{x}$ and $\boldsymbol{\theta}$; and vectors $(\boldsymbol{\lambda}_{\boldsymbol{\theta}}, \boldsymbol{\mu}_{\boldsymbol{\theta}})$ collect the optimal dual variables corresponding to the equality and inequality constraints, respectively.

To save on running time and computational resources, rather than solving $(P_{\boldsymbol{\theta}})$, one can adopt a *learning-to-optimize approach* and train a learning model such as a DNN to predict approximate solutions of $(P_{\boldsymbol{\theta}})$; see e.g., [32]. Once presented with an instance of $\boldsymbol{\theta}$, this DNN can be trained to output a predictor $\hat{\mathbf{x}}(\boldsymbol{\theta}; \mathbf{w})$ of $\mathbf{x}_{\boldsymbol{\theta}}$. The DNN is parameterized by weights $\mathbf{w}$, which can be selected upon minimizing a suitable distance metric or loss function between $\mathbf{x}_{\boldsymbol{\theta}}$ and $\hat{\mathbf{x}}(\boldsymbol{\theta}; \mathbf{w})$ over a training set.

Given a choice for a DNN architecture, the straightforward approach for learning-to-optimize entails two steps:

S1) Building a labeled training dataset $\{\boldsymbol{\theta}_s, \mathbf{x}_{\boldsymbol{\theta}_s}\}_{s=1}^S$ by solving S instances of $(P_{\boldsymbol{\theta}})$; and

S2) Learning $\mathbf{w}$ by minimizing a data fitting loss over the training dataset as

$$\min_{\mathbf{w}} \sum_{s=1}^S \ell\left(\hat{\mathbf{x}}(\boldsymbol{\theta}_s; \mathbf{w}), \mathbf{x}_{\boldsymbol{\theta}_s}\right). \tag{1}$$

For a regression task such as the one considered here, commonly used loss functions ℓ include the mean squared error (MSE) $\|\hat{\mathbf{x}}(\boldsymbol{\theta}; \mathbf{w}) - \mathbf{x}_{\boldsymbol{\theta}}\|_2^2$, or the mean absolute error (MAE) $\|\hat{\mathbf{x}}(\boldsymbol{\theta}; \mathbf{w}) - \mathbf{x}_{\boldsymbol{\theta}}\|_1$. We refer to a DNN trained by solving (1) as a *plain DNN or P-DNN* for short. The conventional P-DNN approach focuses merely on the dataset $\{\boldsymbol{\theta}_s, \mathbf{x}_{\boldsymbol{\theta}_s}\}_{s=1}^S$, and is oblivious of any additional properties the mapping $\boldsymbol{\theta} \to \mathbf{x}_{\boldsymbol{\theta}}$ induced by $(P_{\boldsymbol{\theta}})$ bears.

The key idea here is to extend each training data pair $(\boldsymbol{\theta}_s, \mathbf{x}_{\boldsymbol{\theta}_s})$ as $(\boldsymbol{\theta}_s, \mathbf{x}_{\boldsymbol{\theta}_s}, \mathbf{J}_{\boldsymbol{\theta}_s})$, where $\mathbf{J}_{\boldsymbol{\theta}_s} := [\nabla_{\boldsymbol{\theta}} \mathbf{x}_{\boldsymbol{\theta}}]_{\boldsymbol{\theta}=\boldsymbol{\theta}_s}$ is the Jacobian matrix carrying the partial derivatives of the minimizer $\mathbf{x}_{\boldsymbol{\theta}_s}$ with respect to $\boldsymbol{\theta}$ evaluated at $\boldsymbol{\theta} = \boldsymbol{\theta}_s$ assuming such sensitivities exist. To incorporate the sensitivity information into DNN training, we propose augmenting the loss function with an additional fitting term as

$$\min_{\mathbf{w}} \sum_{s=1}^S \|\hat{\mathbf{x}}(\boldsymbol{\theta}_s; \mathbf{w}) - \mathbf{x}_{\boldsymbol{\theta}_s}\|_2^2 + \rho\|\hat{\mathbf{J}}(\boldsymbol{\theta}_s; \mathbf{w}) - \mathbf{J}_{\boldsymbol{\theta}_s}\|_F^2 \tag{2}$$

where $\rho > 0$ is a scalar weight and $\|\cdot\|_F$ denotes the matrix Frobenius norm. The DNN trained by solving (2) aims to match not only the target output $\mathbf{x}_{\boldsymbol{\theta}}$, but also the sensitivities of $\mathbf{x}_{\boldsymbol{\theta}}$ with respect to $\boldsymbol{\theta}$. We term this neural network a *sensitivity-informed DNN or SI-DNN*.

To better understand (2) and motivate the inclusion of sensitivities, let us put forth the ensuing interpretation. Consider learning function $x : \mathbb{R} \to \mathbb{R}$, which can be an OPF mapping $x(\theta)$ with $P = N = 1$. Under the typical learning setup, one aims to build a DNN $\hat{x}(\theta)$ and approximate $x(\theta)$ given training samples $\{(\theta_s, x(\theta_s))\}_{s=1}^S$. Different from this setup, suppose we are given the additional information of function derivative values at the training samples, so that the training dataset consists of the triplets $\{(\theta_s, x(\theta_s), x'(\theta_s))\}_{s=1}^S$. The pertinent question is how to utilize the extra sensitivity information.

Linearizing function x around a sample θ_s yields

$$x(\theta_s + \epsilon) \simeq x(\theta_s) + \epsilon \cdot x'(\theta_s)$$

for any small ϵ. Linearizing the DNN output yields similarly

$$\hat{x}(\theta_s + \epsilon) \simeq \hat{x}(\theta_s) + \epsilon \cdot \hat{x}'(\theta_s)$$

where $\hat{x}'(\theta_s)$ is the derivative of the DNN output with respect to its input θ evaluated at θ_s. Suppose now ϵ is a zero-mean random variable with variance $\mathbb{E}[\epsilon^2] = \sigma^2$. Instead of training the DNN by minimizing the loss $(\hat{x}(\theta_s) - x(\theta_s))^2$ summed up over all s, one can aim at fitting the function around a sphere of essential radius σ that is centered at θ_s by minimizing

$$\mathbb{E}_\epsilon \left[(\hat{x}(\theta_s + \epsilon) - x(\theta_s + \epsilon))^2 \right]$$
$$\simeq (\hat{x}(\theta_s) - x(\theta_s))^2 + \sigma^2 (\hat{x}'(\theta_s) - x'(\theta_s))^2.$$

Of course, the previous loss is also summed up over all s. Interestingly, this stochastic interpretation of function fitting yields the sensitivity-aware training task of (2) upon identifying $\rho = \sigma^2$. This scalar case of $P = N = 1$ can be trivially extended to a general (OPF) mapping of arbitrary dimensions P and N upon substituting ϵ by a zero-mean random vector $\boldsymbol{\epsilon} \in \mathbb{R}^P$ with covariance matrix $\mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top] = \sigma^2 \mathbf{I}_P$ and replacing derivatives with Jacobian matrices. This interpretation not only justifies the form of (2), but also explains geometrically how sensitivity-informed training uses the point information $\{(\theta_s, x(\theta_s), x'(\theta_s))\}_{s=1}^S$ to extrapolate in a neighborhood around each training datum.

SI-DNNs for learning optimizers were first introduced for solving multiparametric QPs (MPQP) in the conference precursor of this work [28]. For MPQPs, the minimizer $\mathbf{x}_{\boldsymbol{\theta}}$ is known to be a piecewise affine function of $\boldsymbol{\theta}$ [33], [34]. Hence, a DNN with rectified linear unit (ReLU) activations is well-motivated as it can describe such mapping. If hypothetically trained to zero training error, this SI-DNN would yield perfectly accurate predictions in a neighborhood of each training datum θ_s. Numerical tests showed improvements of 2-3 orders of magnitude for SI-DNN over P-DNN in inferring MPQP solutions [28].

This work advocates that the sample efficiency benefit of SI-DNN over P-DNN goes well beyond MPQPs. Before delving into the details and for the sake of visualization, we present some numerical tests on a toy 5-bus power system; a proper numerical evaluation of SI-DNN is deferred to Section VI. This PJM 5-bus system was dispatched via AC-OPF by varying the active load demands $\boldsymbol{\theta} := [p_2^d \ p_4^d]^\top$ on buses 2 and 4 within [1.5, 3.75] per unit (pu) and [0.4, 6] pu, respectively. Fixing the other demands, we dispatched the generators at buses 1 and 5, and removed other generators. Figure 1 depicts the performance improvement of SI-DNN over P-DNN in predicting p_5^g, the optimal dispatch at bus 5.

It is worth clarifying that this work does not train a DNN to predict OPF solutions under *different* power system topologies. On the contrary, it aims at learning the OPF mapping for a single given topology under diverse loading conditions. The fact that the network topology may be changing across time justifies the need to improve on data efficiency, so that after a topology change, the corresponding DNN can be trained afresh using fewer OPF examples generated using the new topology.

This new learning-to-optimize approach alters the two steps of the P-DNN workflow as follows: For step *S1)*, in addition to the minimizer $\mathbf{x}_{\boldsymbol{\theta}_s}$, we now have to compute the Jacobian matrix $\mathbf{J}_{\boldsymbol{\theta}_s}$ for all instances s, if such sensitivities exist. Sections IV and V explain how and when such Jacobian
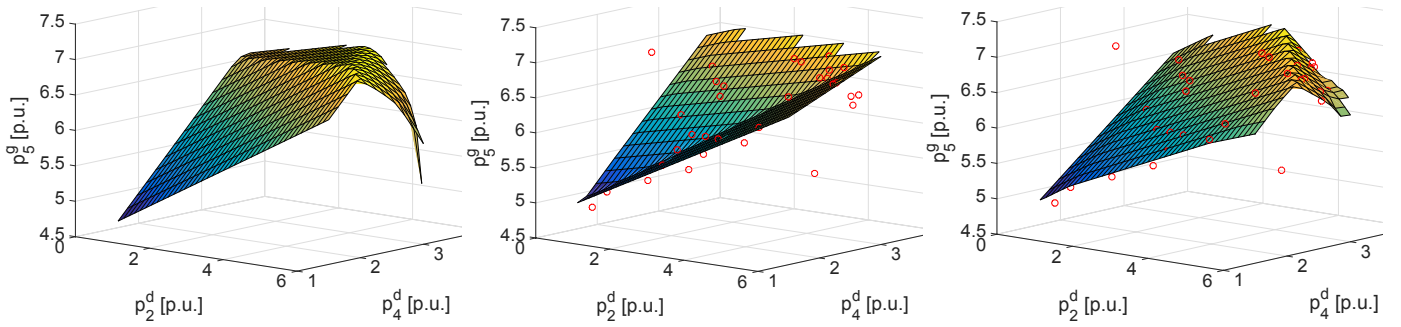
Fig. 1. The left panel depicts the optimal generation dispatch $p_5^g(\boldsymbol{\theta})$ for bus 5 as a function of load demands at buses 2 and 4, that is $\boldsymbol{\theta} := [p_2^d \; p_4^d]^\top$. Sampling the parameter space of $\boldsymbol{\theta}$'s provided 523 feasible OPF instances, of which 37 instances constituted the training set. The center and right panel show the dispatches learned by a P-DNN and an SI-DNN. For the two DNNs, we used the same training points (red circles), architecture (two hidden layers with 16 neurons each), optimizer, and learning rates. P-DNN fails to learn the drop in p_5^g for larger load demands.

matrices can be computed for a non-convex and a convexified rendition of the AC-OPF. The punchline is that obtaining $\mathbf{J}_\theta$ requires minimal additional computational effort and no intervention to the OPF solver. Once the primal/dual solutions have been found by the OPF, computing $\mathbf{J}_\theta$ is as simple as solving a linear system of equations.

For step *S2)*, we migrate from solving (1) to (2). Matrix $\mathbf{J}_{\theta_s}$ is a constant that has been evaluated for each s during *S1)*. Matrix $\hat{\mathbf{J}}(\boldsymbol{\theta}_s; \mathbf{w})$ on the other hand is a function of $\mathbf{w}$ and is not straightforward to compute. Fortunately, computing $\hat{\mathbf{J}}(\boldsymbol{\theta}_s; \mathbf{w})$ can be performed efficiently thanks to advances in automatic differentiation [35]. Modeling (2) in existing DNN software platforms (e.g., TensorFlow) is almost as easy as modeling (1) modulo the coding modifications deferred to Appendix A. Our tests of Section VI further show that the extra computational time for solving (2) is modest. Before computing $\mathbf{J}_\theta$, we first pose AC-OPF as a parametric QCQP.

III. AC-OPF as a Parametric QCQP

A power network with N_b buses can be represented by an undirected connected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, whose nodes $n \in \mathcal{N} := \{1, \dots, N_b\}$ correspond to buses, and edges $e = (n, k) \in \mathcal{E}$ to transmission lines, with cardinality $|\mathcal{E}| := E$. Given line impedances, one can derive the $N_b \times N_b$ bus admittance matrix $\mathbf{Y} = \mathbf{G} + j\mathbf{B}$. Let $v_n = v_n^r + jv_n^i$ and $p_n + jq_n$ denote respectively the complex voltage and power injection at bus $n \in \mathcal{N}$. Power injections are quadratically related to bus voltages through the power flow equations

$$p_n = \sum_{k=1}^{N_b} v_n^r(v_k^r G_{nk} - v_k^i B_{nk}) + v_n^i(v_k^i G_{nk} + v_k^r B_{nk})$$

$$q_n = \sum_{k=1}^{N_b} v_n^i(v_k^r G_{nk} - v_k^i B_{nk}) - v_n^r(v_k^i G_{nk} + v_k^r B_{nk}).$$

If $\mathbf{v} \in \mathbb{R}^{2N_b}$ collects the real and imaginary parts of nodal voltages as $\mathbf{v} := [(\mathbf{v}^r)^\top \; (\mathbf{v}^i)^\top]^\top$ with $\mathbf{v}^r := \{v_n^r\}_{n=1}^{N_b}$ and $\mathbf{v}^i := \{v_n^i\}_{n=1}^{N_b}$, the power flow equations can be written as

$$p_n = \mathbf{v}^\top \mathbf{M}_{p_n} \mathbf{v} \tag{3a}$$

$$q_n = \mathbf{v}^\top \mathbf{M}_{q_n} \mathbf{v} \tag{3b}$$

where $\mathbf{M}_{p_n}$ and $\mathbf{M}_{q_n}$ are $2N_b \times 2N_b$ symmetric real-valued matrices [36]. Squared voltage magnitudes can also be expressed as quadratic functions of $\mathbf{v}$ as

$$|v_n^r + jv_n^i|^2 = \mathbf{v}^\top \mathbf{M}_{v_n} \mathbf{v} \tag{4}$$

where $\mathbf{M}_{v_n} := \mathbf{e}_n \mathbf{e}_n^\top + \mathbf{e}_{N_b+n} \mathbf{e}_{N_b+n}^\top$. The same holds true for the squared magnitude of line currents. If y_{mn} is the series admittance of line $(m, n) \in \mathcal{E}$, the current flowing on this line is $\tilde{i}_{mn} = (\tilde{v}_m - \tilde{v}_n)y_{mn}$, and thus,

$$|\tilde{i}_{mn}|^2 = \mathbf{v}^\top \mathbf{M}_{i_{mn}} \mathbf{v} \tag{5}$$

where $\mathbf{M}_{i_{mn}} := |y_{mn}|^2(\mathbf{e}_m - \mathbf{e}_n)(\mathbf{e}_m - \mathbf{e}_n)^\top + |y_{mn}|^2(\mathbf{e}_{N_b+m} - \mathbf{e}_{N_b+n})(\mathbf{e}_{N_b+m} - \mathbf{e}_{N_b+n})^\top$.

The active power injected into bus n can be decomposed into a dispatchable component p_n^g and an inflexible component p_n^d as $p_n = p_n^g - p_n^d$. The former captures the active power dispatch of a generator or a flexible load located at bus n. The latter captures the inelastic load to be served at bus n. To simplify the exposition, each bus is assumed to be hosting at most one dispatchable resource (generator or flexible load). The reactive power injected into bus n is decomposed similarly as $q_n = q_n^g - q_n^d$. Let $\mathcal{N}_g \subseteq \mathcal{N}$ be the subset of buses hosting dispatchable power injections with cardinality N_g. Bus $n = 1$ belongs to $\mathcal{N}_g$ and serves as the angle reference, so that $v_1^i = \mathbf{v}^\top \mathbf{e}_{N_b+1} \mathbf{e}_{N_b+1}^\top \mathbf{v} = 0$. The remaining buses host non-flexible loads and constitute the subset $\mathcal{N}_\ell = \mathcal{N} \setminus \mathcal{N}_g$ with cardinality $N_l = N_b - N_g$. For simplicity, we will henceforth term the buses in $\mathcal{N}_g$ as *generator buses*, and the ones in $\mathcal{N}_\ell$ as *load buses*. Zero-injection (junction) buses belong to $\mathcal{N}_\ell$ and satisfy $p_n^g = p_n^d = q_n^g = q_n^d = 0$.

Given the inflexible loads at all buses $\{p_n^d, q_n^d\}_{n \in \mathcal{N}}$, the OPF problem aims at optimally dispatching generators and flexible loads $\{p_n^g, q_n^g\}_{n \in \mathcal{N}_g}$ while meeting resource and network limits. The OPF can be formulated as the QCQP [3], [4]

$$\min \quad \sum_{n \in \mathcal{N}_g} c_n^p p_n^g + c_n^q q_n^g \tag{P1}$$

$$\text{over} \quad \mathbf{v} \in \mathbb{R}^{2N_b}, \{p_n^g, q_n^g\}_{n \in \mathcal{N}_g}$$

$$\text{s.to} \quad \mathbf{v}^\top \mathbf{M}_{p_n} \mathbf{v} = p_n^g - p_n^d, \qquad \forall \, n \in \mathcal{N}_g \tag{6a}$$

$$\mathbf{v}^\top \mathbf{M}_{q_n} \mathbf{v} = q_n^g - q_n^d, \qquad \forall \, n \in \mathcal{N}_g \tag{6b}$$

$$\mathbf{v}^\top \mathbf{M}_{p_n} \mathbf{v} = -p_n^d, \qquad \forall \, n \in \mathcal{N}_\ell \tag{6c}$$

$$\mathbf{v}^\top \mathbf{M}_{q_n} \mathbf{v} = -q_n^d, \qquad \forall\, n \in \mathcal{N}_\ell \quad (6d)$$

$$\underline{p}_n^g \leq \mathbf{v}^\top \mathbf{M}_{p_n} \mathbf{v} + p_n^d \leq \bar{p}_n^g, \qquad \forall\, n \in \mathcal{N}_g \quad (6e)$$

$$\underline{q}_n^g \leq \mathbf{v}^\top \mathbf{M}_{q_n} \mathbf{v} + q_n^d \leq \bar{q}_n^g, \qquad \forall\, n \in \mathcal{N}_g \quad (6f)$$

$$\underline{v}_n \leq \mathbf{v}^\top \mathbf{M}_{v_n} \mathbf{v} \leq \bar{v}_n, \qquad \forall\, n \in \mathcal{N} \quad (6g)$$

$$\mathbf{v}^\top \mathbf{e}_{N+1} \mathbf{e}_{N+1}^\top \mathbf{v} = 0 \qquad (6h)$$

$$\mathbf{v}^\top \mathbf{M}_{i_{mn}} \mathbf{v} \leq \bar{i}_{mn}, \qquad \forall\, (m,n) \in \mathcal{E} \quad (6i)$$

where (c_n^p, c_n^q) are the coefficients for generation cost or the utility function for flexible load at bus n. Constraints (6a)–(6d) enforce the power flow equations at load and generator buses. Constraints (6e)–(6f) impose limits for generators and flexible loads. Constraints (6g) confine squared voltage magnitudes within given ranges and (6h) identifies the reference bus. Finally, constraint (6i) limits squared current magnitudes according to line ratings.

Problem (6) is a parametric QCQP as it needs to be solved for different values of demands $\{p_n^d, q_n^d\}_{n \in \mathcal{N}}$; costs $\{c_n\}_{n \in \mathcal{N}_g}$; and generation capacities $\{\underline{p}_n^g, \bar{p}_n^g, \underline{q}_n^g, \bar{q}_n^g\}$. Voltage limits $\{\underline{v}_n, \bar{v}_n\}$ for $n \in \mathcal{N}$ and current limits $\{\bar{i}_{(m,n)}\}$ for $(m,n) \in \mathcal{E}$ may also be changing due to normal and emergency ratings. To keep the exposition uncluttered, we henceforth fix all but the inelastic demands to known values. In other words, we are interested in solving (6) over different values of the parameter vector $\boldsymbol{\theta} := \{p_n^d, q_n^d\}_{n \in \mathcal{N}} \in \mathbb{R}^{2L}$.

The optimization variables of (6) consist of all nodal voltages $\mathbf{v}$ and the (re)active power schedules for generators. If vector $\mathbf{x}_g$ collects generator schedules $\{p_n^g, q_n^g\}_{n \in \mathcal{N}_g}$, then vector $\mathbf{x}_{\boldsymbol{\theta}}^\top := [\mathbf{v}^\top\ \mathbf{x}_g^\top]$ denotes the minimizer of (6) for the specific parameter vector $\boldsymbol{\theta}$. Aiming for the complete $\mathbf{x}_{\boldsymbol{\theta}}$ is apparently an over-parameterization of the problem, adopted only to ease the formulation in (6). What the system operator actually needs to know in practice is only the voltage magnitude and active power schedule for each generator (modulo the reference generator for which we set the voltage magnitude and angle). In light of this and to reduce the DNN output dimension, the DNN is trained to predict the PV setpoints for generators. Given the predicted quantities and knowing the values for inflexible loads from $\boldsymbol{\theta}$, the remaining quantities can be readily computed using a power flow solver anyway.

Given a set of (locally) optimal primal/dual solutions for (6), we next analyze the sensitivity of the parametric AC-OPF. Sensitivities are computed for the complete $\mathbf{x}_{\boldsymbol{\theta}}$, from which the Jacobian $\mathbf{J}_{\boldsymbol{\theta}}$ needed in (2) can be readily obtained.

IV. SENSITIVITY ANALYSIS FOR QCQP-BASED OPF

To analyze the sensitivity of (6) with respect to $\boldsymbol{\theta}$, let us first express the vectors of complex power injections across all buses as

$$\begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} = \mathbf{A}\mathbf{x}_g + \mathbf{B}\boldsymbol{\theta} \qquad (7)$$

where $\mathbf{x}_g$ stacks the generator (re)active power injections $\{p_n^g, q_n^g\}_{n \in \mathcal{N}_g}$ and $(\mathbf{A}, \mathbf{B})$ are matrices assigning generators and loads to buses. We then reformulate (6) as

$$\min_{\mathbf{v}, \mathbf{x}_g}\ \mathbf{a}_0^\top \mathbf{x}_g \qquad (8a)$$

$$\text{s.to}\ \ \mathbf{v}^\top \mathbf{L}_\ell \mathbf{v} = \mathbf{a}_\ell^\top \mathbf{x}_g + \mathbf{b}_\ell^\top \boldsymbol{\theta}, \qquad \ell = 1:L: \quad \lambda_\ell \quad (8b)$$

$$\mathbf{v}^\top \mathbf{M}_m \mathbf{v} \leq \mathbf{d}_m^\top \boldsymbol{\theta} + f_m, \qquad m = 1:M: \quad \mu_m \quad (8c)$$

where $\mathbf{a}_0$ collects the generation cost coefficients (cf. (6)); the first $L = 2N_b + 1$ constraints in (8b) correspond to the power flow equations (6a)–(6d) and the angle reference constraint (6h), while constraints (8c) correspond to the $M = 4N_g + 2N_b + E$ inequality constraints of (6). Matrices $(\mathbf{L}_\ell, \mathbf{M}_m)$ are drawn from the $\mathbf{M}$ matrices appearing in the quadratic forms of (6). Vectors $(\mathbf{a}_\ell, \mathbf{b}_\ell)$ correspond to rows of matrices $(\mathbf{A}, \mathbf{B})$ in (7) for $\ell \leq 2N_b$; and $\mathbf{0}$ for $\ell = 2N_b + 1$. Vectors $\mathbf{d}_m$ are indicator (canonical) vectors and constants f_m relate to generation, voltage, and line limits.

Aiming at computing the sensitivity of a minimizer $\mathbf{x}_{\boldsymbol{\theta}}^\top = [\mathbf{v}^\top\ \mathbf{x}_g^\top]$ of (8) with respect to $\boldsymbol{\theta}$, we explored the related literature. There has indeed been significant interest in computing the sensitivities of OPF minimizers with respect to load [37], [38], [39]. However, the primary motivation for these works was to efficiently compute minimizers and look into binding constraints for a *given trajectory* of load variations. Hence the related OPF was parameterized using a scalar conveniently varied over a range of interest. Seeking to compute the minimizer sensitivities with respect to the vector $\boldsymbol{\theta}$ in a relatively general setting, we explored beyond the power systems literature. Fortunately, there exists a rich corpus of work on perturbation analysis of continuous optimization problems with applications in operation research, economics, mechanics, and optimal control [29]. The first approaches applied the implicit function theorem to the related first-order optimality conditions [30]. Thereon, many developments have been made towards relaxing the assumptions of initial works, and expanding the scope to conic programs [31], [40], [29], [41]. For several recent applications however, the early approaches of [30] are well suited due to their simplicity; see for example [42]. Building upon [30], we next compute the sensitivities required for SI-DNN in Section IV-A; and relax some of the needed assumptions in Section IV-B.

A. Perturbing Optimal Primal/Dual Solutions

Towards instantiating $(P_{\boldsymbol{\theta}})$ with (8) and to reduce notational clutter, let us use symbols $(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ to denote the *optimal* primal/dual variables $(\mathbf{x}_{\boldsymbol{\theta}}, \boldsymbol{\lambda}_{\boldsymbol{\theta}}, \boldsymbol{\mu}_{\boldsymbol{\theta}})$ of (8) for a particular $\boldsymbol{\theta}$. Under mild technical assumptions, a local primal/dual point for (8) satisfies the first-order optimality conditions [31]. The goal of sensitivity analysis is to find infinitesimal changes $(\mathrm{d}\mathbf{x}, \mathrm{d}\boldsymbol{\lambda}, \mathrm{d}\boldsymbol{\mu})$, so that the perturbed point $(\mathbf{x}+\mathrm{d}\mathbf{x}, \boldsymbol{\lambda}+\mathrm{d}\boldsymbol{\lambda}, \boldsymbol{\mu}+\mathrm{d}\boldsymbol{\mu})$ still satisfies the first-order optimality conditions when the input parameters change from $\boldsymbol{\theta}$ to $\boldsymbol{\theta} + \mathrm{d}\boldsymbol{\theta}$ [30]. To this end, we next review the optimality conditions and then differentiate them to compute the sought sensitivities.

The Lagrangian function of (8) is defined as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}; \boldsymbol{\theta}) := \mathbf{a}_0^\top \mathbf{x}_g + \sum_{\ell=1}^{L} \lambda_\ell \left(\mathbf{v}^\top \mathbf{L}_\ell \mathbf{v} - \mathbf{a}_\ell^\top \mathbf{x}_g - \mathbf{b}_\ell^\top \boldsymbol{\theta} \right)$$
$$+ \sum_{m=1}^{M} \mu_m \left(\mathbf{v}^\top \mathbf{M}_m \mathbf{v} - \mathbf{d}_m^\top \boldsymbol{\theta} - f_m \right).$$

With $\mathbf{x} := \{\mathbf{v}, \mathbf{x}_g\}$, Lagrangian optimality $\nabla_{\mathbf{x}}\mathcal{L} = \mathbf{0}$ gives

$$\underbrace{\left(\sum_{\ell=1}^{L} \lambda_\ell \mathbf{L}_\ell + \sum_{m=1}^{M} \mu_m \mathbf{M}_m\right)}_{:=\mathbf{Z}} \mathbf{v} = \mathbf{0}. \tag{9a}$$

$$\mathbf{a}_0 = \sum_{\ell=1}^{L} \lambda_\ell \mathbf{a}_\ell \tag{9b}$$

In addition to Lagrangian optimality, first-order optimality conditions include primal feasibility [cf. (8b)–(8c)], as well as complementary slackness and dual feasibility for all m:

$$\mu_m \underbrace{\left(\mathbf{v}^\top \mathbf{M}_m \mathbf{v} - \mathbf{d}_m^\top \boldsymbol{\theta} - f_m\right)}_{:=g_m} = 0 \tag{10a}$$

$$\mu_m \geq 0. \tag{10b}$$

From the aforementioned optimality conditions, let us focus on those that take the form of equalities, namely (9a)–(9b), (8b), and (10a). For these conditions, we will compute their total differentials. From the first three, we obtain

$$\mathbf{Z}\,\mathrm{d}\mathbf{v} + \mathbf{L}_\lambda\,\mathrm{d}\boldsymbol{\lambda} + \mathbf{M}_\mu\,\mathrm{d}\boldsymbol{\mu} = \mathbf{0} \tag{11a}$$

$$\mathbf{A}^\top\,\mathrm{d}\boldsymbol{\lambda} = \mathbf{0} \tag{11b}$$

$$2\mathbf{L}_\lambda^\top\,\mathrm{d}\mathbf{v} - \mathbf{A}\,\mathrm{d}\mathbf{x}_g - \mathbf{B}\,\mathrm{d}\boldsymbol{\theta} = \mathbf{0} \tag{11c}$$

where $\mathbf{L}_\lambda := \sum_{\ell=1}^{L} \mathbf{L}_\ell \mathbf{v}\mathbf{e}_\ell^\top$; and $\mathbf{M}_\mu := \sum_{m=1}^{M} \mathbf{M}_m \mathbf{v}\mathbf{e}_m^\top$.

For (10a), the total differential is

$$g_m\,\mathrm{d}\mu_m + \mu_m\,\mathrm{d}g_m = 0 \tag{12}$$

where $\mathrm{d}g_m := (\nabla_{\mathbf{v}} g_m)^\top \mathrm{d}\mathbf{v} + (\nabla_{\boldsymbol{\theta}} g_m)^\top \mathrm{d}\boldsymbol{\theta}$ for all m. We identify three cases:

c1) If $\mu_m = 0$ and $g_m < 0$, then (12) implies $\mathrm{d}\mu_m = 0$. It follows that: i) $\mu_m + \mathrm{d}\mu_m = 0$; ii) $(\mu_m + \mathrm{d}\mu_m)(g_m + \mathrm{d}g_m) = 0$; and iii) $g_m + \mathrm{d}g_m < 0$ for any small $\mathrm{d}g_m$. In conclusion, condition (12) ensures that the perturbed point satisfies conditions for optimality, including the inequalities from primal/dual feasibility.

c2) If $\mu_m > 0$ and $g_m = 0$, then (12) gives $\mathrm{d}g_m = 0$. It also follows that: i) $g_m + \mathrm{d}g_m = 0$; ii) $(\mu_m + \mathrm{d}\mu_m)(g_m + \mathrm{d}g_m) = 0$; and iii) $\mu_m + \mathrm{d}\mu_m > 0$ for any small $\mathrm{d}\mu_m$. As in case c1), condition (12) ensures that the perturbed point satisfies all conditions for optimality.

c3) If $\mu_m = g_m = 0$, then (12) is inconclusive on $\mathrm{d}g_m$ and $\mathrm{d}\mu_m$. In this *degenerate* case, for the perturbed point to remain optimal, we need to explicitly impose: i) $\mathrm{d}g_m \leq 0$; ii) $\mathrm{d}\mu_m \geq 0$; and iii) $\mathrm{d}g_m\,\mathrm{d}\mu_m = 0$. Even though the three latter constraints can be handled by the sensitivity analysis of [31], [40], they considerably complicate the treatment. Moreover, such degeneracy is seldom encountered numerically. We henceforth rely on the so called *strict complementarity* assumption, which ignores case c3) [30].

Assumption 1. *Given a tuple of optimal primal/dual variables $(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$, constraint $g_m(\mathbf{x}; \boldsymbol{\theta}) = 0$ if and only if $\mu_m > 0$.*

Two observations are in order. First, the analysis under c1)-c2) reveals that although we perturbed only the equality conditions for optimality, the obtained perturbed point satisfies

the inequality conditions for optimality as well. Therefore, under Assumption 1, the point $(\mathbf{x} + \mathrm{d}\mathbf{x}, \boldsymbol{\lambda} + \mathrm{d}\boldsymbol{\lambda}, \boldsymbol{\mu} + \mathrm{d}\boldsymbol{\mu})$ satisfying the perturbed optimality conditions is (locally) optimal for (8), when solved for $\boldsymbol{\theta} + \mathrm{d}\boldsymbol{\theta}$. Second, despite Assumption 1, if a degenerate instance of (8) does occur for some $\boldsymbol{\theta}_s$ in the training dataset, the particular pair $(\boldsymbol{\theta}_s, \mathbf{x}_{\boldsymbol{\theta}_s})$ can still be used to train the SI-DNN, yet without the additional sensitivity information. In other words, degenerate instances can contribute only to the first fitting term of (2).

Applying (12) for all m, the total derivatives for (10a) can be compactly expressed as

$$\mathcal{D}(\mathbf{g})\,\mathrm{d}\boldsymbol{\mu} + 2\mathcal{D}(\boldsymbol{\mu})\mathbf{M}_\mu^\top\,\mathrm{d}\mathbf{v} - \mathbf{D}^\top\,\mathrm{d}\boldsymbol{\theta} = \mathbf{0}, \tag{13}$$

where $\mathbf{g} := \{g_m\}_{m=1}^{M}$ stacks the inequality constraint values, and matrix $\mathbf{D} := \sum_{m=1}^{M} \mu_m \mathbf{d}_m \mathbf{e}_m^\top$. Operator $\mathcal{D}(\mathbf{x})$ returns a diagonal matrix with vector $\mathbf{x}$ on its main diagonal. Conditions (11) and (13) can be collected in matrix-vector form as

$$\underbrace{\begin{bmatrix} \mathbf{Z} & \mathbf{0} & \mathbf{L}_\lambda & \mathbf{M}_\mu \\ \mathbf{0} & \mathbf{0} & \mathbf{A}^\top & \mathbf{0} \\ 2\mathbf{L}_\lambda^\top & -\mathbf{A} & \mathbf{0} & \mathbf{0} \\ 2\mathcal{D}(\boldsymbol{\mu})\mathbf{M}_\mu^\top & \mathbf{0} & \mathbf{0} & \mathcal{D}(\mathbf{g}) \end{bmatrix}}_{:=\mathbf{S}} \begin{bmatrix} \mathrm{d}\mathbf{v} \\ \mathrm{d}\mathbf{x}_g \\ \mathrm{d}\boldsymbol{\lambda} \\ \mathrm{d}\boldsymbol{\mu} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{B} \\ \mathbf{D}^\top \end{bmatrix}}_{:=\mathbf{U}} \mathrm{d}\boldsymbol{\theta} \tag{14}$$

To compute the sensitivities of primal and dual variables with respect to the p-th entry θ_p of $\boldsymbol{\theta}$, we need to solve the previous system of $2(N_b + N_g) + L + M$ linear equations for $\mathrm{d}\boldsymbol{\theta} = \mathbf{e}_p$. The size of the system can be reduced by dropping the numerous inactive inequality constraints of (8) for which $\mu_m = 0$ and $g_m < 0$, and thus, $\mathrm{d}\mu_m = 0$ as discussed earlier under case c1). Notably, if matrix $\mathbf{S}$ is invertible, the aforementioned sensitivities can all be found at once using the respective blocks of $\mathbf{S}^{-1}\mathbf{U}\mathbf{e}_p$. We next address two relevant questions: *q1) When is $\mathbf{S}$ invertible?*; and *q2) What are the implications of a singular $\mathbf{S}$?*

B. Existence of Primal Sensitivities

To address *q1)* for an arbitrary $(P_{\boldsymbol{\theta}})$, the existing literature identifies some assumptions on $(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}; \boldsymbol{\theta})$. We first review these assumptions, and then assess if they are reasonable for the OPF task at hand. Given an optimal primal $\mathbf{x}$ for some $\boldsymbol{\theta}$, let $\mathcal{A}(\mathbf{x})$ denote the subset of inequality constraints of $\mathbf{g}(\mathbf{x}; \boldsymbol{\theta}) \leq \mathbf{0}$ that are *active or binding*, that is $\mathcal{A}(\mathbf{x}) := \{m : g_m(\mathbf{x}; \boldsymbol{\theta}) = 0\}$. A primal solution $\mathbf{x}$ is termed *regular* if the next assumption holds.

Assumption 2. *The vectors $\{\nabla_{\mathbf{x}} h_\ell\}_{\forall \ell}$ and $\{\nabla_{\mathbf{x}} g_m\}_{m \in \mathcal{A}(\mathbf{x})}$ are linearly independent.*

For the OPF in (8), the functions h_ℓ and g_m correspond to the (in)equality constraints (8b)–(8c) written in the standard form as in $(P_{\boldsymbol{\theta}})$. Assumption 2 is often referred to as linearly independent constraint qualification (LICQ). If a (locally) optimal $\mathbf{x}$ satisfies the LICQ, the corresponding optimal dual variables $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ are known to be unique [43]. In addition to satisfying first-order optimality conditions, a sufficient condition for $(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}; \boldsymbol{\theta})$ to be (locally) optimal is often provided by the following second-order optimality condition.

Assumption 3. *For a subspace orthogonal to the subspace spanned by the gradients of active constraints*

$$\mathcal{Z} := \left\{ \mathbf{z} : \mathbf{z}^\top \nabla_\mathbf{x} h_\ell = 0 \ \forall \ \ell, \mathbf{z}^\top \nabla_\mathbf{x} g_m = 0 \ \forall \ m \in \mathcal{A}(\mathbf{x}) \right\}$$

it holds that $\mathbf{z}^\top \nabla^2_{\mathbf{xx}} \mathcal{L} \mathbf{z} > 0$ for all $z \in \mathcal{Z} \setminus \{\mathbf{0}\}$.

Under the strict complementarity, regularity, and second-order optimality conditions, matrix $\mathbf{S}$ is guaranteed to be invertible; see Theorem 2.1 and Corollary 2.1 of [30].

Lemma 1 ([30]). *If Assumptions 1–3 hold, matrix $\mathbf{S}^{-1}$ exists.*

Lemma 1 implies that under the stated assumptions, the optimal primal and dual variables of $(P_{\boldsymbol{\theta}})$ vary smoothly with changes in parameter $\boldsymbol{\theta}$, and the associated sensitivities can be found via (14). Prior works that compute sensitivities of optimal primal and dual variables for scalar-parameterized OPF instances rely on the non-singularity of $\mathbf{S}$ [37], [38].

While Assumptions 1–3 seem to be standard in the optimization literature, our recent work on the optimal dispatch of inverters in distribution grids demonstrated analytically and numerically that LICQ (Assumption 2) is violated frequently [28]. Instances violating LICQ can be conceived for the AC-OPF in (8) too [44], [45]. To bring up one such example, consider a power system where a load bus m is connected to the rest of the system through another bus n via a single transmission line (m, n). As bus m is a load bus, it contributes two equality constraints (6c) and (6d). It can be shown that if any of the three following scenarios occurs, LICQ fails: *i)* the voltage limits in (6g) become binding (above or below) for both buses m and n; *ii)* line (m, n) becomes congested [cf. (6i)] and a voltage limit at bus m becomes binding; or *iii)* line (m, n) becomes congested and a voltage limit at bus n becomes binding. Further detailed examples for AC-OPF instances violating LICQ can be found in [44], [45]. Attempting to circumvent LICQ violation via problem reformulations may be futile as their occurrences depend on $\boldsymbol{\theta}$, and are thus hard to analyze. Under certain assumptions on OPF instances and load variations, LICQ occurrences can be shown to have zero measure [45]. If the required assumptions are not met, resorting to Fritz-John rather than the KKT conditions for sensitivity analysis has been proposed [44]. However, before tackling the singularity of $\mathbf{S}$ due to LICQ violation, we must answer question *q2)*.

The implications of a singular $\mathbf{S}$ have previously been investigated in [31] and [40]: When LICQ is violated despite strict complementarity, the sensitivities of some primal/dual variables may still exist with respect to a θ_p. In detail, consider the set $\Gamma := \{ \boldsymbol{\gamma} \in \mathbb{R}^{2(N_b + N_g) + L + M} : \mathbf{S}\boldsymbol{\gamma} = \mathbf{U}e_p \}$, which is the solution set of (14). If the n-th entry of $\boldsymbol{\gamma}$ remains constant for all $\boldsymbol{\gamma} \in \Gamma$, the sensitivity of the n-th entry of $[\mathbf{x}^\top \boldsymbol{\lambda}^\top \boldsymbol{\mu}^\top]^\top$ with respect to θ_p does exist; see [31] and [40] for physical interpretation and illustrative examples. While a subset of optimal primal/dual variables may be differentiable under LICQ violation, explicitly identifying the differentiable quantities requires instance-based numerical evaluation in [31] and [40]. Since for training an SI-DNN, we are interested only in the sensitivities $\nabla_{\boldsymbol{\theta}} \mathbf{x}$, we need to ensure that all solutions $\boldsymbol{\gamma} \in \Gamma$ share the same first N entries. This is equivalent to

saying that the first entries of $\mathbf{n}$ are zero for all $\mathbf{n} \in \text{null}(\mathbf{S})$. The equivalence stems from the fact that if $\mathbf{S}\bar{\boldsymbol{\gamma}} = \mathbf{u}$ for a $\bar{\boldsymbol{\gamma}}$, any other solution to $\mathbf{S}\boldsymbol{\gamma} = \mathbf{u}$ takes the form $\boldsymbol{\gamma} = \bar{\boldsymbol{\gamma}} + \mathbf{n}$ for some $\mathbf{n} \in \text{null}(\mathbf{S})$. The next claim provides sufficient conditions for the first N entries of $\mathbf{n}$ to be zero.

Theorem 1. *If Assumptions 1 and 3 hold, then $n_i = 0$ for $i = 1, \ldots, 2(N_b + N_g)$ for all $\mathbf{n} \in \text{null}(\mathbf{S})$.*

Proof: The claim holds trivially for $\mathbf{n} = \mathbf{0}$. The proof for non-zero $\mathbf{n}$ builds on Assumption 3, and thus the terms involved in these assumptions are computed for (8) first.

$$\nabla^2_{\mathbf{xx}} \mathcal{L} := \begin{bmatrix} \mathbf{Z} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \nabla_\mathbf{x} h_\ell := \begin{bmatrix} 2\mathbf{L}_\ell \mathbf{v} \\ -\mathbf{a}_\ell \end{bmatrix}, \nabla_\mathbf{x} g_m := \begin{bmatrix} 2\mathbf{M}_m \mathbf{v} \\ \mathbf{0} \end{bmatrix}.$$

Given vector $\mathbf{n} \neq \mathbf{0}$ such that $\mathbf{S}\mathbf{n} = \mathbf{0}$, partition $\mathbf{n}$ conformably to $[\mathbf{v}^\top \ \mathbf{x}_g^\top \ \boldsymbol{\lambda}^\top \ \boldsymbol{\mu}^\top]^\top$ as $\mathbf{n} := [\mathbf{n}_v^\top \ \mathbf{n}_{x_g}^\top \ \mathbf{n}_\lambda^\top \ \mathbf{n}_\mu^\top]^\top$. By the definition of $\mathbf{S}$ in (14), expanding $\mathbf{S}\mathbf{n} = \mathbf{0}$ yields

$$\mathbf{Z}\mathbf{n}_v + \mathbf{L}_\lambda \mathbf{n}_\lambda + \mathbf{M}_\mu \mathbf{n}_\mu = \mathbf{0} \tag{15a}$$

$$\mathbf{A}^\top \mathbf{n}_\lambda = \mathbf{0} \tag{15b}$$

$$2\mathbf{L}_\lambda^\top \mathbf{n}_v - \mathbf{A}\mathbf{n}_{x_g} = \mathbf{0} \tag{15c}$$

$$2\mathcal{D}(\boldsymbol{\mu})\mathbf{M}_\mu^\top \mathbf{n}_v + \mathcal{D}(\mathbf{g})\mathbf{n}_\mu = \mathbf{0}. \tag{15d}$$

Assumption 1 dictates that the first and second term in (15d) have complementary sparsity, so that $\mathcal{D}(\boldsymbol{\mu})\mathbf{M}_\mu^\top \mathbf{n}_v = \mathbf{0}$ and $\mathcal{D}(\mathbf{g})\mathbf{n}_\mu = \mathbf{0}$. Recalling the definition $\mathbf{M}_\mu := \sum_{m=1}^M \mathbf{M}_m \mathbf{v} e_m^\top$, equation $\mathcal{D}(\boldsymbol{\mu})\mathbf{M}_\mu^\top \mathbf{n}_v = \mathbf{0}$ indeed implies $[\mathbf{n}_v^\top \ \mathbf{n}_{x_g}^\top]\nabla_\mathbf{x} g_m = 0$ for all $m \in \mathcal{A}(\mathbf{x})$, and together with (15c) ensures

$$\begin{bmatrix} \mathbf{n}_v \\ \mathbf{n}_{x_g} \end{bmatrix} \in \mathcal{Z}. \tag{16}$$

Pre-multiplying (15a)–(15b) by $2\mathbf{n}_v^\top$ and $\mathbf{n}_{x_g}^\top$ and subtracting the two resulting equations yields

$$2\mathbf{n}_v^\top \mathbf{Z}\mathbf{n}_v + 2\mathbf{n}_v^\top \mathbf{L}_\lambda \mathbf{n}_\lambda - \mathbf{n}_{x_g}^\top \mathbf{A}^\top \mathbf{n}_\lambda$$
$$+ [\mathbf{n}_v^\top \ \mathbf{n}_x^\top] \begin{bmatrix} 2\mathbf{M}_\mu \\ \mathbf{0} \end{bmatrix} \mathbf{n}_\mu = 0 \tag{17}$$

where the second and third term on the left-hand side (LHS) sum up to zero per (15c). Since $\mathcal{D}(\mathbf{g})\mathbf{n}_\mu = \mathbf{0}$, if $g_m \neq 0$, then the m-th entry $n_{\mu,m}$ of $\mathbf{n}_\mu$ should be zero. In other words, we get that $n_{\mu,m} = 0$ for all $m \notin \mathcal{A}(\mathbf{x})$, and thus, $2[\mathbf{M}_\mu^\top \ \mathbf{0}]^\top \mathbf{n}_\mu = \sum_{m \in \mathcal{A}(\mathbf{x})} \nabla_\mathbf{x} g_m n_{\mu,m}$. Substituting the latter into (17) gives

$$2\mathbf{n}_v^\top \mathbf{Z}\mathbf{n}_v + \sum_{m \in \mathcal{A}(\mathbf{x})} [\mathbf{n}_v^\top \ \mathbf{n}_{x_g}^\top]^\top \nabla_\mathbf{x} g_m n_{\mu,m} = 0.$$

The second term on the LHS equals zero due to (16). Therefore $\mathbf{n}_v^\top \mathbf{Z}\mathbf{n}_v = 0$, thus implying

$$\begin{bmatrix} \mathbf{n}_v^\top & \mathbf{n}_{x_g}^\top \end{bmatrix} \nabla^2_{xx} \mathcal{L} \begin{bmatrix} \mathbf{n}_v \\ \mathbf{n}_{x_g} \end{bmatrix} = 0$$

which contradicts Assumption 3, unless $\mathbf{n}_v = \mathbf{0}$ and $\mathbf{n}_{x_g} = \mathbf{0}$. ∎

Thanks to Theorem 1, we can proceed with computing $\nabla_{\boldsymbol{\theta}} \mathbf{x}$ by solving (14) even if $\mathbf{S}$ is singular. In other words, Theorem 1 allows us to compute $\nabla_{\boldsymbol{\theta}} \mathbf{x}$ even if the LICQ (Assumption 2) fails. If $\mathbf{S}^\dagger$ is the pseudo-inverse of $\mathbf{S}$, the

Jacobian matrix $\mathbf{J_\theta} = \nabla_\theta \mathbf{x}$ can be computed as the top $2(N_b + N_g)$ rows of $-\mathbf{S}^\dagger \mathbf{U}$.

The previous analysis has tacitly presumed the system $\mathbf{S}\boldsymbol{\gamma} = \mathbf{u}$ has at least one solution for all $\mathbf{u} \in \text{range}(\mathbf{U})$. The numerical tests of Section VI demonstrate that for the AC-OPF in (8), the system $\mathbf{S}\boldsymbol{\gamma} = \mathbf{u}$ features a solution indeed.

As discussed earlier, we focus on training an SI-DNN for predicting generator voltage magnitudes and active power setpoints. Having solved (14) and found the sensitivity of $\mathbf{x}$ with respect to $\boldsymbol{\theta}$, the sensitivity of active power generation can be obtained readily using the corresponding entries of $\mathbf{x}_g$. The sensitivity of voltage magnitudes can be derived from the sensitivities of the real and imaginary components of voltages with respect to $\boldsymbol{\theta}$. Precisely, the voltage magnitude at bus n is given by $v_n = \sqrt{(v_n^r)^2 + (v_n^i)^2}$ and its sensitivity with respect to θ_ℓ can be found through the chain rule

$$\frac{\partial v_n}{\partial \theta_\ell} = \frac{1}{v_n}\left(v_n^r \frac{\partial v_n^r}{\partial \theta_\ell} + v_n^i \frac{\partial v_n^i}{\partial \theta_\ell}\right).$$

Evaluating the above completes the requirements of sensitivities for augmenting the SI-DNN training set.

From (14), the required sensitivities obviously depend on values of optimal primal/dual variables of (8). Problem (8) is a non-convex quadratic program and existing solvers may converge to a local rather than a global solution. Albeit the previous sensitivity analysis is valid even for local solutions, the performance of the trained DNN will be apparently sub-optimal. To train an SI-DNN to predict global OPF solutions, we next extend the analysis to the SDP relaxation of (8).

V. SDP Relaxation of the AC-OPF

In pursuit of globally optimal AC-OPF schedules, the non-convex QCQP of (8) can be relaxed to the SDP [2]

$$\min_{\mathbf{x}_g, \mathbf{V} \succeq 0} \mathbf{a}_0^\top \mathbf{x}_g \tag{18a}$$

$$\text{s.to} \quad \text{Tr}(\mathbf{L}_\ell \mathbf{V}) = \mathbf{a}_\ell^\top \mathbf{x}_g + \mathbf{b}_\ell^\top \boldsymbol{\theta}, \qquad \forall \ell: \ \lambda_\ell \tag{18b}$$

$$\text{Tr}(\mathbf{M}_m \mathbf{V}) \leq \mathbf{d}_m^\top \boldsymbol{\theta} + f_m, \qquad \forall m: \ \mu_m. \tag{18c}$$

Problem (18) is equivalent to (8) if matrix $\mathbf{V}$ is rank-1 at optimality, in which case the SDP relaxation is deemed as *exact*. The relaxation turns out to be exact for several power networks and practical loading conditions; see [5] for a review of related analyses. When the relaxation is exact, the $\mathbf{V}$ minimizer of (18) can be expressed as $\mathbf{V} = \mathbf{v}\mathbf{v}^\top$.

We briefly review how the solution $(\mathbf{x}_g, \mathbf{v}; \boldsymbol{\lambda}, \boldsymbol{\mu})$ obtained from the SDP formulation of (18) satisfies the first-order optimality conditions for the non-convex QCQP in (8) as well. To this end, it is not hard to derive the dual program of (18):

$$\max_{\boldsymbol{\lambda}, \boldsymbol{\mu}} -\sum_{\ell=1}^{L} \lambda_\ell \mathbf{b}_\ell^\top \boldsymbol{\theta} - \sum_{m=1}^{M} \mu_m (\mathbf{d}_m^\top \boldsymbol{\theta} + f_m) \tag{19a}$$

$$\text{s.to} \quad \mathbf{a}_0 = \sum_{\ell=1}^{L} \lambda_\ell \mathbf{a}_\ell \tag{19b}$$

$$\mathbf{Z} := \sum_{\ell=1}^{L} \lambda_\ell \mathbf{L}_\ell + \sum_{m=1}^{M} \mu_m \mathbf{M}_m \succeq 0 \tag{19c}$$

$$\boldsymbol{\mu} \geq \mathbf{0}. \tag{19d}$$

The optimality conditions for the SDP primal-dual pair (18)–(19) then include:

i) Primal feasibility (18b)–(18c) implies (8b)–(8c).
ii) Dual feasibility (19d) applies to (8) as well.
iii) Complementary slackness for (18c) applies to (8c).
iv) Complementary slackness for (19c) gives $\text{Tr}(\mathbf{V}\mathbf{Z}) = 0$ or $\mathbf{Z}\mathbf{v} = \mathbf{0}$, which along with (19b) yield the Lagrangian optimality conditions for the QCQP shown in (9).

Therefore $(\mathbf{x}_g, \mathbf{v}; \boldsymbol{\lambda}, \boldsymbol{\mu})$ is a stationary point for the QCQP in (8). Because it further attains the optimal cost for the relaxed problem in (18), it is in fact the globally optimal for (8).

To recapitulate, we have used the non-convex QCQP formulation of the AC-OPF to derive the sensitivity formulae of (14). This is advantageous as the QCQP features differentiable objective and constraint functions. The obtained sensitivity formulae can be evaluated at the AC-OPF solution provided by any nonlinear programming solver, although such solution may be only locally optimal. To compute a globally optimum AC-OPF solution, we propose using (18) instead. If the SDP relaxation is exact, the obtained solution is globally optimal, while the sensitivity formulae derived from QCQP can still be used. Our suggested workflow avoids computing the sensitivities of the SDP formulation for the AC-OPF: Even though differentiating through convex cone constraints is possible [41], it can be perplexing.

Remark 1. *The aforesaid workflow runs the SDP-based solver to obtain an OPF solution, but computes its sensitivities using the convenient formulae associated with the QCQP-OPF. This is to ensure global optimality if the SDP relaxation is exact. An alternative way to check global optimality is to follow the workflow of [46]: Obtain an OPF solution via a mature OPF solver (e.g., QCQP or MATPOWER), and use the optimality conditions of the SDP-based OPF to check whether the obtained QCQP- or MATPOWER-based solution is globally optimal. Nevertheless, this optimality check relies on sufficient conditions. As a result, if the QCQP- or MATPOWER-based solution does not pass the global optimality test (that is indeed the case for the IEEE 300-bus system [46]), one may still have to run the SDP-based OPF solver in pursuit of a better solution or a global optimality guarantee.*

VI. Numerical Tests

The novel SI-DNN approach was evaluated using the IEEE 39-bus, the IEEE 118-bus, and the Illinois 200-bus system. Datasets were generated using either the nonlinear OPF MATPOWER or the globally optimal SDP-based solver.

A. DNN Architecture and Training

To ease the implementation and without loss of generality, we assumed that buses hosting generators do not host loads, i.e., $p_n^d = q_n^d = 0$ for all $n \in \mathcal{N}_g$. As discussed at the end of Section III, the DNN input $\boldsymbol{\theta}$ consists of the $2(N_b - N_g)$ (re)active power demands at load buses. The DNN output is the setpoints for active power and voltage magnitude (p_n^g, v_n)

at all generators $n \in \mathcal{N}_g$ excluding p_1^g for the slack bus. We collect these output quantities in $\check{\mathbf{x}}_{\boldsymbol{\theta}}$, a subvector of $\mathbf{x}_{\boldsymbol{\theta}}$.

Both for P-DNN and SI-DNN, we chose a feed-forward fully-connected architecture. For the number of hidden layers being K, denote the number of neurons in layer k by u_k, with input dimension $u_0 = 2(N_b - N_g)$ and output dimension $u_{K+1} = 2N_g - 1$. To explicitly constrain DNN outputs as per (6e) and (6g), the output layer uses $\tanh$ as its activation function, while all other layers use ReLU. For DNN training and evaluation, labels $\check{\mathbf{x}}_{\boldsymbol{\theta}}$ were suitably scaled within $[-1, 1]$.

We built all DNNs using the `TensorFlow 2.0` python platform alongside `Keras` libraries. Training an SI-DNN deviates from the default routine as gradient updates are implemented separately; see Appendix A for key differences. For DNN training, at every weight-update step, the gradients computed via the procedure in the appendix are passed to the Adam optimizer. For all tests, optimizer Adam was used with an exponential decay reducing the rate to 85% every 250 epochs. The initial learning rate will be reported later. DNNs were compiled using Jupyter notebook on a 2.7 GHz Intel Core i5 computer with 8 GB RAM.

B. Learning Locally Optimal OPF Solutions

We first trained DNNs towards predicting MATPOWER AC-OPF minimizers. We contrasted SI-DNN with P-DNN in terms of the MSE and the related training times. With the primary goal of improving sample efficiency, the numerical tests emphasize on performance evaluation for relatively small training datasets. Nevertheless, to gain insight on the effect of the training dataset size, we first present tests using larger training datasets.

1) Tests on IEEE 39-bus system with large training datasets: The network parameters and nominal loads for the IEEE 39-bus system were fetched from MATPOWER `casefile` [1]. The 39-bus system hosts $N_g = 10$ generators. The benchmark system has loads on two of the generator buses. Removing these, there are 29 load buses. To build a dataset for DNN testing and training, a set of 12,000 random $\boldsymbol{\theta} \in \mathbb{R}^{2 \cdot 29}$ was sampled. The corresponding $\check{\mathbf{x}}_{\boldsymbol{\theta}}$'s were obtained via MATPOWER. The dataset thus obtained was partitioned into training, cross-validation, and testing sets of sizes 10,000; 1,000; and 1,000, respectively. If infeasibility is encountered for some $\boldsymbol{\theta}$'s, such instances were omitted from the dataset. To represent various demand levels, we sampled the 12,000 random $\boldsymbol{\theta}$'s by scaling the benchmark demands entry-wise by a scalar drawn independently and uniformly within $[0.8, 1.2]$. For the aforementioned sampling, all 12,000 OPF instances were feasible. Since the generator cost functions are identical in the benchmark system, a uniform active power cost was used for all generators. The default OPF formulation of MAT-POWER deviates from the QCQP in (P1). These differences introduce some nuances in building the linear system of (14) for computing sensitivities; see Appendix B for details. Having built the aforementioned dataset $\{(\boldsymbol{\theta}_s, \mathbf{J}_{\boldsymbol{\theta}_s}, \check{\mathbf{x}}_{\boldsymbol{\theta}_s})\}_{s=1}^{12000}$, the architectures for SI-DNN and P-DNN were determined next. Based on preliminary tests, identical architectures were chosen for P-DNN and SI-DNN with $K = 4$ hidden layers with

TABLE I
AVERAGE TEST MSE [$\times 10^{-3}$] FOR PREDICTING MATPOWER
SOLUTION ON IEEE 39-BUS SYSTEM, AND WEIGHTING FACTOR ρ

| Training | MSE | | |
Size	**P-DNN**	**SI-DNN**	ρ
100	2.80	1.50	10
1000	1.00	0.59	2
5000	0.54	0.32	1
10000	0.19	0.14	0.2

TABLE II
AVERAGE TEST MSE [$\times 10^{-3}$] AND TRAINING TIME [IN SEC] FOR
PREDICTING MATPOWER SOLUTION ON IEEE 39-BUS SYSTEM

| Training | **P-DNN** | | **SI-DNN** | |
Size	MSE	Time	MSE	Time
10	8.6	738	3.3	746
50	4.3	739	2.1	756
100	3.2	747	2.0	776
250	1.9	302	2.0	332

$u_k = 256$ neurons for $k = 1, \ldots, 4$. Preliminary tests showed negligible effect on P-DNN performance if the number of layers is reduced to three. Nevertheless, the architecture for the two DNNs was kept identical to ensure equal expressibility.

The performance of the two DNNs was evaluated in terms of the MSE for training sizes $(100, 1000, 5000, 10000)$ sampled from the complete training set of size 10000. The batch-size for all tests was fixed to 100. The cross-validation set was used to determine the initial learning rate (ILR), epochs needed, and the factor ρ in (2). The ILR for training sizes $(100, 1000, 5000, 10000)$ was $(5, 5, 10, 50) \times 10^{-4}$ and the epochs needed were $(2000, 500, 500, 250)$ for both DNNs. The decrease in the training epochs needed is due to the increase in gradient steps per epoch for larger training sizes with fixed batch size. The MSEs obtained by the two DNNs averaged over the 1000 test instances are provided in Table I alongside the factor ρ used for different training sizes. As anticipated, the test errors for both DNNs decrease for larger training sizes. However, the SI-DNN consistently outperforms the P-DNN with the improvement being more pronounced at smaller training sizes. Interestingly, a decreasing trend in the suitable choice of ρ was obtained from cross-validation indicating that as the training samples become abundant, sensitivity information seems to be becoming less important. The remaining numerical tests explicitly focus on small training sizes. For simplicity, hereon we fix the ILR to 5×10^{-4} and $\rho = 20$.

2) Tests on IEEE 39-bus system with small training datasets: A dataset $\{(\boldsymbol{\theta}_s, \mathbf{J}_{\boldsymbol{\theta}_s}, \check{\mathbf{x}}_{\boldsymbol{\theta}_s})\}_{s=1}^{1000}$ was created following the methodology delineated in the previous subsection to evaluate the two DNNs when the training sizes are varied over a data-scarce regime. The evaluation was performed as follows. First, for a *training size* of 10, we created 20 different training sets by sampling 10 OPF instances from the dataset without replacement. For each of these 20 times or *runs*, the OPF instances not sampled for training consisted the testing sets. We then separately trained P-DNN and SI-DNN on these 20 sets. For the training sizes of $(50, 100, 250)$, we had $(20, 10, 4)$ runs, respectively. For training sizes $(10, 50, 100)$, the entire training set was used for gradient computation at
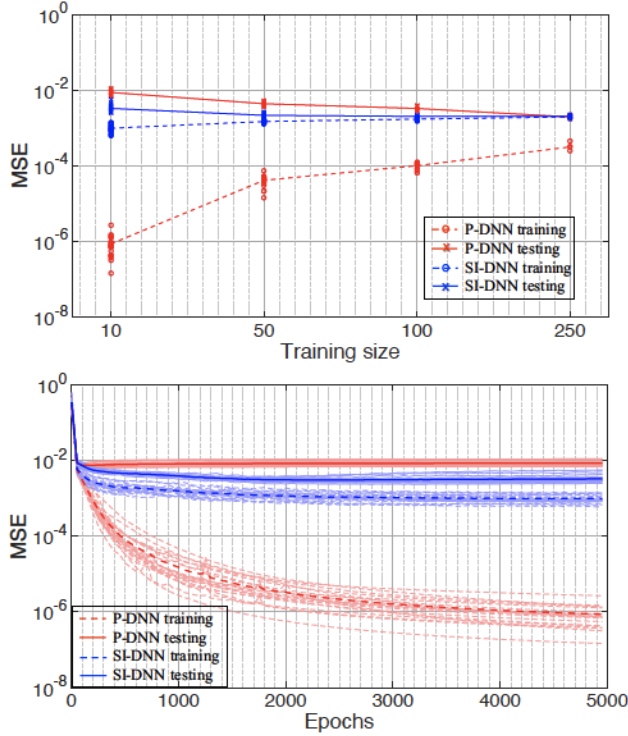
Fig. 2. Average training and testing errors for different training sizes (*top*); and errors across epochs for different runs with training size 10 (*bottom*).

TABLE III
AVERAGE TEST MSE [$\times 10^{-3}$] AND TRAINING TIME [IN SEC] FOR PREDICTING MATPOWER SOLUTION

Train. Size	IEEE 118-bus				Illinois 200-bus			
	P-DNN		SI-DNN		P-DNN		SI-DNN	
	MSE	Time	MSE	Time	MSE	Time	MSE	Time
25	1.8	447	1.1	483	0.19	452	0.04	491
50	1.7	458	1.1	527	0.15	456	0.04	524
100	1.6	463	0.9	610	0.09	471	0.06	608

each step, with the total epochs being 5000. When the training size was 250, the batch-size was fixed to 100, and total epochs to 2000. The training and testing MSE loss for all training sizes, and runs are shown in Fig. 2 (*top*). For the tests with training size 10, the evolution of DNN errors are shown in Fig. 2 (*bottom*). The average test MSE and training times for the two DNNs are shown in Table II. From Fig. 2 (*top*), we observe as anticipated, that for both DNNs, the gap between training and testing loss decreases for larger training size. Further, the errors for different *runs* are well clustered, indicating a numerically stable DNN implementation. From Table II, it is fascinating to note that the test loss attained by SI-DNN is much lower than P-DNN, especially at smaller training sets. For instance, the P-DNN requires 100 samples to roughly attain the average test MSE which the SI-DNN attains with 10 samples. The lower MSE for P-DNN with training size 250 is a repercussion of not updating ρ for varying training sizes, which was avoided for simplicity. It is worth stressing that the improvement in sample efficiency comes at modest increase in training time.

3) Tests on other benchmarks: The DNN architecture chosen for the other two power systems was similar to the IEEE 39-bus case with the differences being in the number

TABLE IV
AVERAGE TIME [IN SEC] TO SOLVE AC-OPF, t_{OPF}; COMPUTE SENSITIVITIES, t_{SA}; AND OBTAIN DNN PREDICTIONS FOLLOWED BY RUNNING AC POWER FLOW, t_{INFER}

Test System	t_{OPF}	t_{sa}	t_{infer}
39-bus	0.1229	0.0034	0.0039
118-bus	0.2577	0.0260	0.0050
200-bus	0.3032	0.0811	0.0078

of neurons per layer. Specifically, the DNNs used for the 118- and 200-bus systems had 512 neurons in hidden layers, with the input (output) layers having 128 (107), and 302 (97) neurons, respectively. For each of these systems, we created a dataset with 500 feasible[1] random demands generated by scaling the nominal demands entry-wise by factors drawn uniformly from [0.7,1.3]. The linear cost coefficients from the respective benchmark systems were retained as c_p's, while the reactive power cost coefficients were set to zero. All DNNs were evaluated for five runs, with training sizes of 25, 50, and 100. Table III summarizes the obtained results.

Having evaluated the improvement in MSE brought by the sensitivity-informed learning approach, we next assessed the additional time-complexity introduced for computing the desired sensitivities. Specifically, while building the datasets for the IEEE 39-bus, the IEEE 118-bus, and the Illinois 200-bus system, we computed: *i)* the average time t_{OPF} taken by MATPOWER to solve an OPF instance; *ii)* the average time t_{sa} required for computing the Jacobian matrix $\mathbf{J}_{\boldsymbol{\theta}_s}$ using (14)[2]; and *iii)* the average time t_{infer} needed to obtain a *complete* OPF minimizer using SI-DNN during the inference phase. To do the latter, we summed up the time taken for evaluating SI-DNN predictions and the time needed to evaluate a corresponding AC power flow solution using MATPOWER. It must be noted that evaluating t_{infer} is merely to assess an approximate speed-up offered by the DNNs over conventional OPF solvers. It does not constitute a rigorous comparison since neither optimality nor feasibility is guaranteed for DNN predictions. The aforementioned times are reported in Table IV. It is exciting to observe that while the SI-DNN approach can reduce the training size requirement by up to a factor of 10, evaluating sensitivities for training the SI-DNN requires substantially less time than solving an OPF instance. Finally, the average speed-up factor $t_{\text{OPF}}/t_{\text{infer}}$ obtained for the 39-, 118-, and 200-bus systems was approximately 34, 63, and 52, respectively.

C. Learning Globally Optimal OPF Solutions

The SI-DNN was evaluated towards predicting the minimizer of an SDP relaxation-based OPF solver for the IEEE 39-bus system. A uniform active power cost was used for all generators while the reactive power cost coefficients were set as $c_n^q = 0.1 c_n^p$. To build a dataset, a set of $1,000$ random θ's was sampled as explained earlier. The corresponding $\check{\mathbf{x}}_\theta$'s

---

[1]To obtain a dataset of 500 instances, the OPF was solved for 550 instances and the first 500 feasible instances were retained. For the 118-bus system, all instances were feasible while for the 200-bus system, four infeasible instances were encountered.

[2]For improved numerical performance, matrix **S** was stored as a sparse matrix and (14) was solved using MATLAB's command `lsqminnorm`.

TABLE V
AVERAGE TEST MSE [$\times 10^{-3}$] FOR PREDICTING SDP SOLUTIONS, AND CONSTRAINT VIOLATION STATISTICS ON THE IEEE 39-BUS SYSTEM

Train. Size	P-DNN				SI-DNN			
	MSE	(a)	(b)	(c)	MSE	(a)	(b)	(c)
10	6.3	2.61	0.50	9.78	0.91	2.52	0.37	3.35
50	3.6	2.45	0.55	7.38	0.62	2.58	0.27	2.06
100	2.5	2.59	0.53	6.87	0.67	2.52	0.27	1.96
(a) #violations /instance; **(b)** max. violation; **(c)** mean violation [$\times 10^{-4}$]								

TABLE VI
AVERAGE TEST MSE [$\times 10^{-4}$] FOR PREDICTING MATPOWER SOLUTION AND TRAINING SET GENERATION TIME-BUDGET t_{GEN}

Benchmark	t_{gen} [sec]	P-DNN MSE	SI-DNN MSE	Improvement %
39-bus	1.2	0.86	0.24	72.1
	6.2	0.49	0.21	57.1
	12.3	0.33	0.19	40.6
	30.7	0.17	0.12	29.4
118-bus	6.45	17.50	10.80	38.3
	12.89	17.84	12.94	27.5
	25.77	17.01	7.82	54.1
200-bus	7.58	1.83	0.32	82.5
	15.16	1.46	0.45	69.2
	30.32	0.91	0.57	37.4

were obtained by solving (18) using the MATLAB-based optimization toolbox YALMIP with SDP solver MOSEK [47]. For all SDP instances, the ratio of the second largest eigenvalue of matrix $\mathbf{V}$ to the largest eigenvalue was found to lie in $[3 \cdot 10^{-7}, 1 \cdot 10^{-4}]$; numerically indicating an exact relaxation. Thus, the eigenvector corresponding to the largest eigenvalue was deemed as the optimal voltage $\mathbf{v}$. If instances with inexact relaxation are encountered, they can be omitted from the dataset. As with learning MATPOWER solutions, the sample efficiency of SI-DNN was found superior to P-DNN in learning globally optimal OPF solutions; see Table V for the average MSE attained during testing. The presented results with local and global OPF solvers demonstrate that SI-DNN yields a dramatic improvement in generalizability. The feasibility statistics included in Table V are elaborated upon in Section VI-E.

D. DNN Performance Evaluation under a Time Budget

Tables I–III and V attest the improved sample efficiency of sensitivity-informed over conventional training under various dataset sizes, benchmark networks, and OPF solvers. This section exemplifies how these results translate to gains in MSE for a fixed time budget. Specifically, P-DNN and SI-DNN were compared when allotted identical times to complete training. This evaluation was carried out for the settings described in Section VI-B. For all tests, the number of epochs for P-DNN was kept fixed as provided in Section VI-B, while the number of epochs for SI-DNN was reduced, so as to match the training time of the P-DNN. Next, a common time budget t_{gen} was fixed for creating the training datasets. Based on Table IV, the training sizes for P-DNN and SI-DNN can be approximately computed as $t_{\text{gen}}/t_{\text{OPF}}$ and $t_{\text{gen}}/(t_{\text{OPF}} + t_{\text{sa}})$; implying smaller training sets for the SI-DNN. The test MSEs obtained for

TABLE VII
(a) AVERAGE VIOLATIONS PER INSTANCE; (b) MAXIMUM VIOLATION; AND (c) MEAN VIOLATION [$\times 10^{-4}$] FOR PREDICTING MATPOWER SOLUTION

Benchmark	Train. Size	P-DNN			SI-DNN		
		(a)	(b)	(c)	(a)	(b)	(c)
39-bus	10	2.29	1.01	15	1.91	0.33	7.66
	50	2.45	0.95	9.55	2.14	0.24	6.30
	100	2.31	0.71	7.65	1.96	0.26	6.26
	250	2.23	0.51	6.99	1.87	0.28	6.89
200-bus	25	10.43	1.42	8.10	4.99	1.37	4.27
	50	9.08	1.19	6.17	3.69	1.14	4.64
	100	10.89	1.15	7.17	2.74	1.24	4.71

the aforementioned setup are provided in Table VI. It was observed that with identical time budgets, an SI-DNN yields test MSEs that are 28-83% less compared to P-DNN.

E. Assessing Feasibility of DNN Predictions

While emphasis has been on MSE, the importance of satisfying constraints cannot be undermined. To this end, we tested the feasibility of SI-DNN OPF predictions using the following metrics. For each of the DNNs, given a test input and the associated DNN prediction, an AC power flow solution was obtained using MATPOWER. For each instance, the inequalities in (8c) not directly enforced by the $\tanh$ activation were evaluated. These included voltage limits on load buses, line flow limits, generator reactive power limits, and the slack bus active power limits, totalling to 126, 424, and 647 constraints for the 39-, 118-, and 200-bus system, respectively. For suitable scaling, the violations in flows and generation were normalized by the maximum limit. To be specific, a normalized violation of 10^{-3} in generator power injection translates to a violation of 0.1% of the maximum power capacity of that generator. Voltage violations were maintained in pu.

We first evaluated the constraint violations caused by SI-DNN and P-DNN predictions while learning globally optimal OPF solutions obtained from the SDP-based solver. The assessment was carried out on the test instances that remained after sampling training sets of different sizes from the 1,000 random instances [cf. Section VI-C]. For different training sizes, Table V lists: *a)* the average number of violations exceeding a normalized magnitude of 10^{-6} per test instance; *b)* the maximum constraint violation observed; and *c)* the violations averaged over all constraints and test instances. Interestingly, while both DNNs incur similar count of violations, SI-DNN reduces the maximum violation by half and the mean violation to less than one third. We further investigated into the specific constraints being violated by SI-DNN predictions. Interestingly, there were just 5 constraints frequently violated. Three of these were minimum reactive power generation, and the remaining were maximum active power of the slack generator and a line flow limit.

We repeated the previous feasibility analysis for the DNNs aimed at learning locally optimal OPF solutions from MATPOWER. The constraint violation statistics obtained for the IEEE 39-bus and the Illinois 200-bus system, provided in Table VII, consistently demonstrate the improvements yielded

TABLE VIII
(a) AVERAGE VIOLATIONS PER INSTANCE; (b) MAXIMUM VIOLATION;
AND (c) MEAN VIOLATION [$\times 10^{-3}$] FOR PREDICTING MATPOWER
SOLUTION ON IEEE 118-BUS SYSTEM

Constraint Set	Train. Size	P-DNN			SI-DNN		
		(a)	(b)	(c)	(a)	(b)	(c)
Full	25	7.97	1.08	1.90	9.28	1.44	3.90
	50	8.24	1.02	1.60	10.20	0.97	3.30
	100	8.45	1.18	1.80	10.09	0.93	2.20
Reduced	25	2.12	0.73	0.32	1.67	0.66	0.14
	50	1.92	0.63	0.21	1.65	0.68	0.14
	100	2.01	1.17	0.34	1.77	0.78	0.12

by the SI-DNN approach for different training sizes. Table VIII reports the same statistics for the IEEE 118-bus system while predicting the MATPOWER solution. The numerical observations for the IEEE 118-bus system do not align with the results for other benchmark networks. The statistics provided in the top part of Table VIII exhibit much higher constraint violations for both P-DNN and SI-DNN; note that the mean violations are of the order 10^{-3} as opposed to 10^{-4} for other networks. Moreover, SI-DNN performs worse than P-DNN on several metrics. Spurred by the exceptionally high constraint violations, we investigated the individual constraints being violated. It was found that several generator reactive power limits were being consistently violated by both P-DNN and SI-DNN. It turns out that these limits were binding for *all* the random OPF scenarios in the training and testing datasets. For benchmarks that exhibit such patterns with certain dispatch quantities being fixed across scenarios, it may be prudent to set them at the respective values and solve a reduced OPF. To emphasize on the violation statistics for the non-trivial constraints, we computed the feasibility metrics on a reduced set of constraints not including those reactive power limits that were consistently binding. The obtained results shown at the bottom of Table VIII corroborate the superior performance of SI-DNN over P-DNN.

VII. CONCLUSIONS

This work has built on the fresh idea of sensitivity-informed training for learning the solutions of arbitrary AC-OPF formulations. It comprehensively delineated the steps for computing the involved sensitivities using the optimal primal/dual solutions, which are readily available by AC-OPF solvers. Such sensitivities of the primal AC-OPF solutions have been shown to exist under mild assumptions, while their computation is as simple as solving a system of linear equations with multiple right-hand sides. The approach is quite general since the OPF solutions comprising the training dataset can be obtained by off-the-shelf nonlinear OPF solvers or modern conic relaxation-based schemes. It is also worth stressing that sensitivity-informed training can readily complement other existing learn-to-OPF methodologies. Extensive numerical tests on three benchmark power systems have demonstrated that with a modest increase in training time, SI-DNNs attain the same prediction performance as conventionally trained DNNs by using roughly only 1/10 to 1/4 of the training data. Such improvement on sample efficiency reduces the time needed for generating training datasets, and

is thus, relevant to delay-critical power systems applications. Furthermore, SI-DNN predictions turn out to feature better constraint satisfaction capabilities too. Sensitivity-informed learning forms the solid foundations for several exciting and practically relevant research directions, such as warm-starting key optimal primal/dual variables to accelerate decentralized OPF solvers and predicting active constraints.

APPENDIX

A. Python Implementation for SI-DNN

A typical implementation example for computing the gradient of the MSE loss in P-DNN with respect to the DNN weights (which are the *trainable variables*) involves

```
with tensorflow.GradientTape() as tape:
    pred_x = model(theta)
    loss = keras.losses.MSE(xlabel,pred_x)
model_gradients=tape.gradient(loss,
            model.trainable_variables)
```

where `model` represents the DNN and `GradientTape` computes the desired gradient. In transitioning to SI-DNN, we first need to compute the gradient of the DNN output `pred_x` with respect to its input `theta` to define the loss. We then compute the gradients of the two loss terms with respect to the DNN weights. This can be implemented using nested `GradientTape` as

```
with tensorflow.GradientTape() as tape:
  with tensorflow.GradientTape() as tape2:
    tape2.watch(theta)
    pred_x=model(theta)
    Ploss=keras.losses.MSE(xlabel,pred_x)
  J_model=tape2.batch_jacobian(pred_x,
                    theta)
  J_flat=tf.keras.backend.reshape(fgrad,
                    shape=(1,))
  SI_loss=keras.losses.MSE(J_flat,J_label)
  total_loss=P_loss+rho*SI_loss
model_gradients=tape.gradient(loss,
            model.trainable_variables)
```

where the inner `tape` computes the sensitivity of DNN to compute the overall SI-DNN loss, while the outer `tape` computes the gradients for weight updates.

B. Sensitivity computation with MATPOWER

While solving the AC-OPF instances with MATPOWER, we used the Cartesian coordinate system, and flow limits were imposed on squared currents. For computing the desired sensitivities, we first need to build the linear system (14), which requires the optimal dual variables, constraint function values, and the derivatives of the constraint functions with respect to the optimization variables. Although MATPOWER can deal with the AC-OPF posed with voltages in Cartesian coordinates, it slightly differs from the QCQP in (6) as follows:

a1) MATPOWER enforces power flow equations as in (6a)–(6d). Different from (6e)–(6f) however, MATPOWER poses generator (re)active power limits as $\underline{p}_n^g \leq p_n^g \leq \bar{p}_n^g$. Thus, the related $\nabla_{\mathbf{v}} g_m$ becomes zero and $\nabla_{\mathbf{x}_g} g_m$ becomes a signed canonical vector corresponding to bus n.

a2) MATPOWER constraints voltages, rather than squared voltages as in (6g). While the two versions are equivalent, the related derivatives $\nabla_{\mathbf{v}} g$ apparently differ. The

derivatives of non-squared magnitudes can be found using the chain rule as $\nabla_{\mathbf{v}} v_n = \frac{\partial v_n}{\partial v_n^2} \nabla_{\mathbf{v}} v_n^2 = \frac{1}{v_n} \nabla_{\mathbf{v}} v_n^2$ and $\nabla_{\mathbf{v}} v_n^2 = 2\mathbf{M}_{v_n}\mathbf{v}$ from (4).

a3) Different from (6h), MATPOWER sets the voltage angle reference to zero by enforcing $\arctan(v_{N+1}^i / v_{N+1}^r) = 0$. Fortunately, simply setting the imaginary part v_{N+1}^i to zero is equivalent, and the gradients of these two formulations agree. Thus, despite the difference in formulation, we use (6h) wherever needed in building (14).

a4) Finally, MATPOWER poses flow limits on both the sending and receiving ends of each line, thus doubling the number of constraints in (6i). The matrices $\mathbf{M}_{\mathbf{i}_{mn}}$ can be built using the *from bus* and *to bus* admittances obtained via the MATPOWER command `makeYbus()`.

REFERENCES

[1] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "MAT-POWER: steady-state operations, planning and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, Feb. 2011.

[2] X. Bai, H. Wei, K. Fujisawa, and Y. Yang, "Semidefinite programming for optimal power flow problems," *Intl. Journal of Electric Power & Energy Systems*, vol. 30, no. 6, pp. 383–392, 2008.

[3] S. Bose, D. Gayme, K. Chandy, and S. Low, "Quadratically constrained quadratic programs on acyclic graphs with application to power flow," *IEEE Trans. Control of Network Systems*, vol. 2, no. 3, pp. 278–287, Sep. 2015.

[4] R. Madani, S. Sojoudi, and J. Lavaei, "Convex relaxation for optimal power flow problem: Mesh networks," *IEEE Trans. Power Syst.*, vol. 30, no. 1, pp. 199–211, Jan. 2015.

[5] S. Low, "Convex relaxation of optimal power flow – Part II: Exactness," *IEEE Trans. Control of Network Systems*, vol. 1, no. 2, pp. 177–189, Jun. 2014.

[6] A. S. Xavier, F. Qiu, and S. Ahmed, "Learning to solve large-scale security-constrained unit commitment problems," *INFORMS Journal on Computing*, pp. 1–18, Oct. 2020, (early access).

[7] F. Fioretto, T. W. Mak, and P. V. Hentenryck, "Predicting AC optimal power flows: Combining deep learning and Lagrangian dual methods," in *AAAI Conf. on Artificial Intelligence*, New York, NY, Feb. 2020.

[8] X. Pan, T. Zhao, and M. Chen, "DeepOPF: Deep neural network for DC optimal power flow," in *Proc. IEEE Intl. Conf. on Smart Grid Commun.*, Beijing, China, Oct. 2019, pp. 1–6.

[9] 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," in *Proc. IEEE Intl. Conf. on Smart Grid Commun.*, Tempe, AZ, Nov. 2020, pp. 1–6.

[10] X. Pan, M. Chen, T. Zhao, and S. H. Low, "Deepopf: A feasibility-optimized deep neural network approach for AC optimal power flow problems," 2020, (preprint). [Online]. Available: https://arxiv.org/abs/2007.01002

[11] N. Guha, Z. Wang, M. Wytock, and A. Majumdar, "Machine learning for AC optimal power flow," 2019, climate Change Workshop at ICML 2019. [Online]. Available: https://arxiv.org/abs/1910.08842

[12] A. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," in *Proc. IEEE Intl. Conf. on Smart Grid Commun.*, Tempe, AZ, Nov. 2020, pp. 1–6.

[13] Y. Zhao and B. Zhang, "Deep learning in power systems," in *Advanced Data Analytics for Power Systems*, A. Tajer, S. M. Perlaza, and H. V. Poor, Eds. Cambridge, UK: Cambridge University Press, May 2021.

[14] D. Owerko, F. Gama, and A. Ribeiro, "Optimal power flow using graph neural networks," in *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Process.*, Barcelona, Spain, May 2020, pp. 5930–5934.

[15] S. Gupta, V. Kekatos, and M. Jin, "Communication-limited inverter control using deep neural networks," in *Proc. IEEE Intl. Conf. on Smart Grid Commun.*, Tempe, AZ, Nov. 2020, pp. 1–6.

[16] H. Lange, B. Chen, M. Berges, and S. Kar, "Learning to solve AC optimal power flow by differentiating through holomorphic embeddings," 2020, (submitted). [Online]. Available: https://arxiv.org/abs/2012.096224

[17] S. Gupta, S. Misra, D. Deka, and V. Kekatos, "DNN-based policies for stochastic AC-OPF," in *Proc. Power Syst. Comput. Conf.*, Porto, Portugal, Jun. 2021, (to appear also in the Elsevier Electric Power Systems Research). [Online]. Available: https://www.faculty.ece.vt.edu/kekatos/papers/PSCC2022a.pdf

[18] S. Gupta, V. Kekatos, and M. Jin, "Controlling smart inverters using proxies: A chance-constrained DNN-based approach," *IEEE Trans. Smart Grid*, May 2021, (submitted). [Online]. Available: https://arxiv.org/abs/2105.00429

[19] Y. Chen, S. Lakshminarayana, C. Maple, and H. V. Poor, "A meta-learning approach to the optimal power flow problem under topology reconfigurations," 2020, (preprint). [Online]. Available: https://arxiv.org/abs/2012.11524

[20] Y. Chen and B. Zhang, "Learning to solve network flow problems via neural decoding," 2020, preprint. [Online]. Available: https://arxiv.org/abs/2002.04091

[21] D. Deka and S. Misra, "Learning for DC-OPF: Classifying active sets using neural nets," in *IEEE PowerTech*, Milan, Italy, Jun. 2019, pp. 1–6.

[22] M. Yatin Nandwani, Abhishek Pathak and P. Singla, "A primal dual formulation for deep learning with constraints," in *Proc. of Adv. Neural Inf. Process. Syst.*, Vancouver, Canada, Dec. 2019, pp. 12 157–12 168.

[23] L. Zhang, Y. Chen, and B. Zhang, "A convex neural network solver for DCOPF with generalization guarantees," 2020, (submitted). [Online]. Available: https://arxiv.org/abs/2009.09109

[24] L. Zhang, G. Wang, and G. B. Giannakis, "Real-time power system state estimation and forecasting via deep unrolled neural networks," *IEEE Trans. Signal Processing*, vol. 67, no. 15, pp. 4069–4077, Aug. 2019.

[25] Q. Yang, A. Sadeghi, G. Wang, G. B. Giannakis, and J. Sun, "Robust PSSE using graph neural networks for data-driven and topology-aware priors," 2020, (submitted). [Online]. Available: https://arxiv.org/abs/2003.01667

[26] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[27] G. S. Misyris, A. Venzke, and S. Chatzivasileiadis, "Physics-informed neural networks for power systems," in *Proc. IEEE PES General Meeting*, Montreal, Canada, Aug. 2020, pp. 1–5.

[28] M. K. Singh, S. Gupta, V. Kekatos, G. Cavraro, and A. Bernstein, "Learning to optimize power distribution grids using sensitivity-informed deep neural networks," in *Proc. IEEE Intl. Conf. on Smart Grid Commun.*, Tempe, AZ, Nov. 2020, pp. 1–6.

[29] J. F. Bonnans and A. Shapiro, *Perturbation Analysis of Optimization Problems*. New York, NY: Springer Science & Business Media, 2000.

[30] A. V. Fiacco, "Sensitivity analysis for nonlinear programming using penalty methods," *Mathematical Programming*, vol. 10, no. 1, pp. 287–311, Dec. 1976.

[31] A. J. Conejo, E. Castillo, R. Minguez, and R. Garcia-Bertrand, *Decomposition Techniques in Mathematical Programming*. Springer, 2006.

[32] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for interference management," *IEEE Trans. Signal Processing*, vol. 66, no. 20, pp. 5438–5453, Oct. 2018.

[33] F. Borrelli, A. Bemporad, and M. Morari, "Geometric algorithm for multiparametric linear programming," *Journal of Optimization Theory and Applications*, vol. 118, no. 3, pp. 515–540, Sep. 2003.

[34] S. Taheri, M. Jalali, V. Kekatos, and L. Tong, "Fast probabilistic hosting capacity analysis for active distribution systems," *IEEE Trans. Smart Grid*, 2020, (early access).

[35] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 5595–5637, Jan. 2017.

[36] V. Kekatos, G. Wang, H. Zhu, and G. B. Giannakis, "PSSE redux: Convex relaxation, decentralized, robust, and dynamic approaches," in *Advances in Power System State Estimation*, M. El-Hawary, Ed. Wiley, 2021.

[37] K. Almeida, F. Galiana, and S. Soares, "A general parametric optimal power flow," *IEEE Trans. Power Syst.*, vol. 9, no. 1, pp. 540–547, Feb. 1994.

[38] V. Ajjarapu and N. Jain, "Optimal continuation power flow," *Electric Power Systems Research*, vol. 35, no. 1, pp. 17–24, Oct. 1995.

[39] K. Almeida and R. Salgado, "Optimal power flow solutions under variable load conditions," *IEEE Trans. Power Syst.*, vol. 15, no. 4, pp. 1204–1211, Nov. 2000.

[40] E. Castillo, A. J. Conejo, C. Castillo, R. Minguez, and D. Ortigosa, "Perturbation approach to sensitivity analysis in mathematical programming," *Journal of Optimization Theory and Applications*, vol. 128, no. 1, pp. 49–74, Jan. 2006.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPWRS.2021.3127189, IEEE Transactions on Power Systems

IEEE TRANSACTIONS ON POWER SYSTEMS (TO APPEAR) 14

[41] A. Agrawal, S. Barratt, S. Boyd, E. Busseti, and W. M. Moursi, "Differentiating through a cone program," 2020, (submitted). [Online]. Available: https://arxiv.org/abs/1904.09043

[42] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," in *Intl. Conf. on Machine Learning*, Sydney, NSW, Australia, 2017, p. 136–145.

[43] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 1999.

[44] K. C. Almeida and A. Kocholik, "Solving ill-posed optimal power flow problems via Fritz-John optimality conditions," *IEEE Trans. Power Syst.*, vol. 31, no. 6, pp. 4913–4922, Nov. 2016.

[45] A. Hauswirth, S. Bolognani, G. Hug, and F. Dorfler, "Generic existence of unique lagrange multipliers in AC optimal power flow," *IEEE Contr. Syst. Lett.*, vol. 2, no. 4, pp. 791–796, Oct. 2018.

[46] D. K. Molzahn, B. C. Lesieutre, and C. L. DeMarco, "A sufficient condition for global optimality of solutions to the optimal power flow problem," *IEEE Trans. Power Syst.*, vol. 29, no. 2, pp. 978–979, Mar. 2014.

[47] J. Lofberg, "Yalmip : A toolbox for modeling and optimization in matlab," in *Proc. of the CACSD Conf.*, Taipei, Taiwan, 2004.

Vassilis Kekatos (SM'16) is an Associate Professor with the Bradley Dept. of ECE at Virginia Tech. He obtained his Diploma, M.Sc., and Ph.D. in computer science and engineering from the Univ. of Patras, Greece, in 2001, 2003, and 2007, respectively. He is a recipient of the NSF Career Award in 2018 and the Marie Curie Fellowship. He has been a research associate with the ECE Dept. at the Univ. of Minnesota, where he received the postdoctoral career development award (honorable mention). During 2014, he stayed with the Univ. of Texas at Austin and the Ohio State Univ. as a visiting researcher. His research focus is on optimization and learning for future energy systems. He is currently serving in the editorial board of the IEEE Trans. on Smart Grid.

Manish K. Singh received the B.Tech. degree from the Indian Institute of Technology (BHU), Varanasi, India, in 2013; and the M.S. and Ph.D. degrees in electrical engineering from Virginia Tech, Blacksburg, VA, USA, in 2018 and 2021, respectively. He is currently a postdoctoral researcher with the University of Minnesota, Minneapolis, MN. During 2013-2016, he worked as an Engineer in the Smart Grid Dept. of POWERGRID, the central transmission utility of India. His research interests are focused on optimization, control, and learning techniques to develop algorithmic solutions for the operation and analysis of electric power systems as well as water and natural gas networks.

Georgios B. Giannakis (Fellow, IEEE) received the Diploma in electrical engineering from the National Technical University of Athens, Athens, Greece, in 1981, the M.Sc. degree in electrical engineering, the M.Sc. degree in mathematics, and the Ph.D. degree in electrical engineering from the University of the Southern California, Los Angeles, CA, USA, in 1983, 1986, and 1986, respectively. He was a faculty member with the University of Virginia, Charlottesville, VA, USA, from 1987 to 1998, and since 1999, he has been a Professor with the University of Minnesota, Minneapolis, MN, USA, where he held an ADC Endowed Chair of telecommunications, was the Director of the Digital Technology Center from 2008 to 2021, and since 2016, he is has been a University of Minnesota McKnight Presidential Chair in ECE.

He has authored or coauthored more than 480 journal papers, 780 conference papers, 25 book chapters, two edited books, and two research monographs in his research field, which include statistical learning, signal processing, communications, and networking. His current research include data science, and network science with applications to the Internet of Things, and power networks with renewables. He is the co-inventor of 34 issued patents. He was the co-recipient of the ten Best Journal Paper Awards from the IEEE Signal Processing (SP) and Communications Societies, including the G. Marconi Prize Paper Award in Wireless Communications. He was also recipient of the IEEE-SPS Norbert Wiener Society Award (2019), the EURASIP's A. Papoulis Society Award (2020), Technical Achievement Awards from the IEEE-SPS (2000) and from EURASIP (2005), the IEEE ComSoc Education Award (2019), and the IEEE Fourier Technical Field Award (2015). He is a Foreign Member of the Academia Europaea, and a Fellow of the National Academy of Inventors, the European Academy of Sciences, and EURASIP. He has served the IEEE in a number of posts, including a Distinguished Lecturer for the IEEE-SPS.