Learning to Optimize Power Distribution Grids using Sensitivity-Informed Deep Neural Networks

Manish K. Singh, Sarthak Gupta, Vassilis Kekatos
Bradley Dept. of ECE, Virginia Tech
Blacksburg, VA, 24061
Emails: {manishks,gsarthak,kekatos}@vt.edu

Guido Cavraro and Andrey Bernstein
National Renewanable Energy Laboratories
Golden, CO, 80401
Emails: {guido.cavraro,Andrey.Bernstein}@nrel.gov

***Abstract*—Deep learning for distribution grid optimization can be advocated as a promising solution for near-optimal yet timely inverter dispatch. The principle is to train a deep neural network (DNN) to predict the solutions of an optimal power flow (OPF), thus shifting the computational effort from real-time to offline. Nonetheless, before training this DNN, one has to solve a large number of OPFs to create a labeled dataset. Granted the latter step can still be prohibitive in time-critical applications, this work puts forth an original technique for improving the prediction accuracy of DNNs by taking into account the sensitivities of the OPF minimizers with respect to the OPF parameters. By expanding on multiparametric programming, it is shown that although inverter control problems may exhibit dual degeneracy, the required sensitivities do exist in general and can be computed readily using the output of any standard quadratic program (QP) solver. Numerical tests showcase that sensitivity-informed deep learning can enhance prediction accuracy in terms of mean square error (MSE) by 2-3 orders of magnitude at minimal computational overhead. Improvements are more significant in the small-data regime, where a DNN has to learn to optimize using a few examples. Beyond multiparametric QPs, the approach is currently being generalized to parametric (non)-convex optimization problems.**

***Index Terms*—Multiparametric programming; reactive power compensation; optimal power flow; linearized distribution flow.**

I. Introduction

To cope with resource allocation problems with stringent latency requirements, recent research advocates using learning models that are trained to solve optimization tasks; see e.g., [1], [2]. The general workflow involves solving a large number of such problems offline to create a dataset of problem instances paired with their minimizers; train a learning model to predict those minimizers; and use the model for promptly finding near-optimal solutions in real time. References [1] and [3] impose feasibility using Lagrangian decomposition, while [4] and [5] build DNNs with structures mimicking algorithmic iterates solving the original optimization.

In the context of power systems, there has recently been an interesting interplay between deep learning and the OPF. DNNs have been employed to predict OPF solutions under a linearized DC [6], [7]; or the exact AC model [3], [8]. The standard regression loss function is sometimes augmented by a penalty on OPF constraint violations to promote feasibility [6], [8], [3]. Alternatively, predicted solutions may be converted to a physically implementable schedule upon projection using a power flow (PF) solver [6], [7]. A graph neural network leveraging the connectivity of the power system is trained to infer AC-OPF solutions in [9]. Focusing on OPF for inverter control, reference [10] constructs a DNN to be used as a digital twin of the electric feeder. To learn optimal inverter control policies, DNN-based reinforcement learning schemes have been suggested in [11], [12].

Instead of learning minimizers, DNNs have also been used to predict the active constraints of an OPF [8], [13], [2]. References [8] and [13] train DNN-based classifiers to predict individual or sets of active constraints. In [2], the active constraints of a linear programming (LP) DC-OPF are unveiled in three steps: A DNN is first trained to predict the optimal generation cost given demands; locational prices are then computed as the DNN sensitivities with respect to demands; and congested lines are identified via sparse-aware regression.

When the OPF is posed using a linearized grid model, it gives rise to parametric LPs or QPs, and hence, powerful tools from multiparametric programming (MPP) become relevant [14]. Such tools have been utilized before to study congestion patterns in electricity markets [15]; for the probabilistic characterization of primal/dual OPF solutions [16]; to accelerate hosting capacity analyses [17]; or strategic investment [18]. MPP-aided methods however are meaningful only when the related OPF enjoys few congestion patterns and/or the application of interest is of a batch nature.

This work contributes to the literature of training DNNs to solve the OPF in the following creative way. A DNN is trained to fit not only the OPF minimizer, but also its sensitivities with respect to the problem parameters. Inspired by [19], we cross-pollinate the idea of training sensitivity-aware DNNs from solving differential equations to learning for optimization problems in Section II. Advances in automatic differentiation facilitate the efficient computation of DNN sensitivities as well as their integration into training. In Sections III and IV, we particularize properties from MPP theory to the inverter dispatch task at hand, so that the sensitivities of the OPF solution can be computed readily. The numerical tests of Section V showcase that the proposed sensitivity-informed DNN outperforms a plain DNN in terms of prediction accuracy by 2-3 orders of magnitude. The paper is concluded with

This work was supported in part by grants NSF-2034137 and NSF-1751085.
978-1-7281-6127-3/20/$31.00

generalizations of the method to more challenging OPF tasks.

Regarding *notation*, lower- (upper-) case boldface letters denote column vectors (matrices). Sets are represented by calligraphic symbols. Symbol $^\top$ stands for transposition, and inequalities are understood element-wise. All-zero and all-one vectors and matrices are represented by $\mathbf{0}$ and $\mathbf{1}$; the respective dimensions are deducible from context. A symmetric positive definite matrix is denoted as $\mathbf{X} \succ \mathbf{0}$.

II. Sensitivity-Informed Deep Learning

Suppose a system operator has to routinely solve a convex quadratic program (QP) over the resource vector $\mathbf{x}$

$$\mathbf{x}_{\boldsymbol{\theta}} := \arg\min_{\mathbf{x}} \ \tfrac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x} - \mathbf{x}^\top \mathbf{B}\boldsymbol{\theta} \tag{1a}$$

$$\text{s.to} \ \ \mathbf{C}\mathbf{x} \leq \mathbf{D}\boldsymbol{\theta} + \mathbf{e} : \quad \boldsymbol{\lambda}_{\boldsymbol{\theta}}. \tag{1b}$$

While $(\mathbf{A} \succ \mathbf{0}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{e})$ are kept fixed for some time, the parameter vector $\boldsymbol{\theta}$ may be varying frequently. Let $(\mathbf{x}_{\boldsymbol{\theta}}, \boldsymbol{\lambda}_{\boldsymbol{\theta}})$ denote a pair of primal/dual solutions for a particular $\boldsymbol{\theta}$. This multiparametric QP (MPQP) may be encountered in several resource allocation tasks in cyber-physical systems in general, and power systems in particular. Renditions of (1) may appear in real-time electricity markets and contingency analysis in transmission systems; or in load disaggregation, demand response, and DER management in distribution systems.

The application of interest here is dispatching inverters through an approximate OPF that minimizes power losses subject to voltage constraints. Vector $\mathbf{x}$ comprises the inverter reactive power setpoints, while $\boldsymbol{\theta}$ carries the loading conditions (demand and solar generation) across feeder buses. Since $\boldsymbol{\theta}$ may change rapidly (rooftop solar generation can fluctuate by up to 15% of its rating within one minute [20]), an operator may not be able to solve (1) exactly for thousands of feeders hosting thousands of inverters each.

To expedite the process of computing inverter setpoints, the operator may want to train a DNN that *learns to solve* (1). This DNN would be fed with $\boldsymbol{\theta}$ to compute a predictor $\hat{\mathbf{x}}(\boldsymbol{\theta}; \mathbf{w})$ of $\mathbf{x}_{\boldsymbol{\theta}}$. The straightforward approach for training this DNN is finding its weights $\mathbf{w}$ as the minimizers of

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

Problem (2) is solved offline and aims at fitting the DNN output to the OPF minimizer over S loading scenarios $\{\boldsymbol{\theta}_s\}_{s=1}^S$, which are deemed representative of the upcoming control period. Before solving (2) however, the operator has to solve S instances of (1) to build the labeled dataset $\{(\boldsymbol{\theta}_s, \mathbf{x}_{\boldsymbol{\theta}_s})\}_{s=1}^S$. Accommodating a large S could be impractical in time-critical setups, where the DNN has to be re-trained every 30–60 min or so because parameters $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{e})$ change too. This is relevant for inverter control due to grid topology reconfigurations, changes in regulator settings and capacitor statuses, and varying loading conditions that may alter the linearization point of the nonlinear power flow equations.

We improve on this process in two aspects: *i)* we reduce S so the operator has to solve fewer OPFs; and *ii)* for each OPF instance solved during training, we further exploit the sensitivity of $\mathbf{x}_s$ with respect to $\boldsymbol{\theta}_s$. Both aspects can expedite training or increase the training dataset over the same training time, while *ii)* is expected to yield better generalization to the sought DNN and stabilize its solutions against variations in $\boldsymbol{\theta}$. Inspired by [19] where a DNN was trained to solve differential equations by incorporating information on its derivatives, here we integrate sensitivity information of the optimizer $\mathbf{x}_{\boldsymbol{\theta}}$ with respect to the DNN input $\boldsymbol{\theta}$. To accomplish that, we propose augmenting the training process of (2) as

$$\min_{\mathbf{w}} \sum_{s=1}^{S} \|\hat{\mathbf{x}}(\boldsymbol{\theta}_s; \mathbf{w}) - \mathbf{x}_{\boldsymbol{\theta}_s}\|_2^2 + \|\hat{\mathbf{J}}(\boldsymbol{\theta}_s; \mathbf{w}) - \mathbf{J}_{\boldsymbol{\theta}_s}\|_F^2 \tag{3}$$

where $\mathbf{J}_{\boldsymbol{\theta}} := \nabla_{\boldsymbol{\theta}} \mathbf{x}_{\boldsymbol{\theta}}$ and $\hat{\mathbf{J}}(\boldsymbol{\theta}; \mathbf{w}) := \nabla_{\boldsymbol{\theta}} \hat{\mathbf{x}}(\boldsymbol{\theta}; \mathbf{w})$ are the Jacobian matrices of the minimizer and the DNN output accordingly with respect to the parameter vector $\boldsymbol{\theta}$. Symbol $\|\cdot\|_F$ denotes the Frobenius norm. Matrix $\hat{\mathbf{J}}(\boldsymbol{\theta}_s; \mathbf{w})$ and its derivatives with respect to $\mathbf{w}$ can be readily computed via automatic differentiation tools such as GradientTape in TensorFlow. The optimization of (3) aims at fitting the values of the minimizers as well as their sensitivities (partial derivatives) in terms of the DNN input $\boldsymbol{\theta}$. By incorporating such physics-aware information, the DNN is expected to predict well not only on the training scenarios $\boldsymbol{\theta}_s$'s, but in a neighborhood around them too. We term the DNN trained by (3) as *sensitivity-informed DNN* (SI-DNN) versus the *plain DNN* (P-DNN) obtained from (2).

Having posed the sensitivity-informed deep learning task of (3), the pertinent questions are if the OPF operator $\text{OPF} : \boldsymbol{\theta} \to \mathbf{x}_{\boldsymbol{\theta}}$ is differentiable; and if so, whether its Jacobian matrix $\mathbf{J}_{\boldsymbol{\theta}}$ can be found in a computationally efficient way. The goal is to be able to build the augmented training set $\{(\boldsymbol{\theta}_s, \mathbf{x}_{\boldsymbol{\theta}_s}, \mathbf{J}_{\boldsymbol{\theta}_s})\}$ at a minimal computational effort in excess to the effort for building the training set $\{(\boldsymbol{\theta}_s, \mathbf{x}_{\boldsymbol{\theta}_s})\}$ for (2). To this end, Section III first poses optimal inverter dispatching in the form of the MPQP in (1), and then Section IV adapts results from multiparametric programming to compute $\mathbf{J}_{\boldsymbol{\theta}}$ efficiently.

III. Optimal Inverter Dispatch

Consider the task of optimal reactive power compensation by smart inverters. Given (re)active loads along with the active solar generation on every bus, the goal is to find the inverter setpoints for reactive power injections. Such setpoints can be found through an OPF that minimizes ohmic power losses on distribution lines, while satisfying voltage regulation limitations across buses. Before formally posing this OPF, we briefly review the postulated feeder model.

In a single-phase radial feeder with $N+1$ buses, the substation is indexed by $n = 0$. Let v_n be the voltage magnitude, and $p_n + jq_n$ the complex power injection at bus n. Collect all but the substation injections and voltages in the N-length vectors $(\mathbf{p}, \mathbf{q}, \mathbf{v})$. Vectors $(\mathbf{p}, \mathbf{q})$ can be decomposed into nonnegative injections $(\mathbf{p}_i, \mathbf{q}_i)$ from inverters, and withdrawals $(\mathbf{p}_\ell, \mathbf{q}_\ell)$ from loads as

$$\mathbf{p} = \mathbf{F}_i \mathbf{p}_i - \mathbf{F}_\ell \mathbf{p}_\ell \tag{4a}$$

$$\mathbf{q} = \mathbf{F}_i \mathbf{q}_i - \mathbf{F}_\ell \mathbf{q}_\ell. \tag{4b}$$

Matrices $\mathbf{F}_i$ and $\mathbf{F}_\ell$ map buses with solar and loads to the N feeder buses, respectively. If there are I buses hosting solar and L buses hosting loads, then $\mathbf{F}_i \in \{0,1\}^{N\times I}$ and $\mathbf{F}_\ell \in \{0,1\}^{N\times L}$. We assume that each bus may host at most one (aggregate) load and at most one solar generator.

To simplify the task of dispatching inverters, we resort to an approximate feeder model obtained upon linearizing the power flow equations at the flat voltage profile of unit magnitudes and zero angles at all buses; see e.g., [17]. According to this model, voltages depend linearly on power injections as

$$\mathbf{v}(\mathbf{p},\mathbf{q}) \simeq \mathbf{R}\mathbf{p} + \mathbf{X}\mathbf{q} + v_0 \mathbf{1}_N \tag{5}$$

where the $N \times N$ matrices $(\mathbf{R}, \mathbf{X})$ are symmetric positive definite, and depend on the feeder topology and line impedances. We are also interested in expressing ohmic losses with respect to nodal injections. Adopting a second-order Taylor's series expansion, active losses on lines can be approximated as a convex quadratic function of power injections as

$$L(\mathbf{p},\mathbf{q}) \simeq 2\mathbf{p}^\top \mathbf{R}\mathbf{p} + 2\mathbf{q}^\top \mathbf{R}\mathbf{q}. \tag{6}$$

Based on the previous modeling, the reactive setpoints for inverters can be found as the minimizers of

$$\min_{\mathbf{q}_i} \quad \mathbf{q}^\top \mathbf{R}\mathbf{q} \tag{7a}$$

$$\text{s.to} \quad -0.03 \cdot \mathbf{1}_N \le \mathbf{R}\mathbf{p} + \mathbf{X}\mathbf{q} \le 0.03 \cdot \mathbf{1}_N \tag{7b}$$

$$-\bar{\mathbf{q}}_i \le \mathbf{q}_i \le \bar{\mathbf{q}}_i. \tag{7c}$$

The approximate OPF of (7) minimizes ohmic losses over the controllable inverter setpoints $\mathbf{q}_i$; note the first summand of (6) and the scaling by 2 have been ignored. Constraint (7b) maintains voltages within $\pm 3\%$ per unit (pu). Although the substation voltage v_0 has been set at 1 pu, it can be appended as an optimization variable of (7) without loss of generality. Our methodology can incorporate voltage regulators as in [17]; they are ignored here to keep the exposition succinct. Inverter setpoints are constrained in (7c) by the vector $\bar{\mathbf{q}}_i$ of reactive ratings, which is assumed known and fixed.

Under varying grid conditions, an operator may want to solve (7) routinely for different OPF instances. If the topology remains fixed, OPF instances differ in the values of loads and solar $(\mathbf{p}_i, \mathbf{p}_\ell, \mathbf{q}_\ell)$. Define the vector of OPF inputs $\boldsymbol{\theta} := [\mathbf{p}^\top \ \mathbf{q}_\ell^\top]^\top$. Due to the problem structure, active loads and solar can appear collectively as $\mathbf{p}$ per (4a). To better distinguish the optimization variable $\mathbf{x} = \mathbf{q}_i$ from the OPF parameters $\boldsymbol{\theta}$, let us substitute (4b) in (7), and express the inverter dispatch of (7) as the MPQP of (1) with the substitutions

$$\mathbf{A} := 2\mathbf{F}_i^\top \mathbf{R}\mathbf{F}_i \succ \mathbf{0}, \qquad \mathbf{B} := 2\mathbf{F}_i^\top \mathbf{R}\mathbf{F}_\ell \, [\mathbf{0}_{L\times N} \ \ \mathbf{I}_L] \tag{8a}$$

$$\mathbf{C} := \begin{bmatrix} -\mathbf{X}\mathbf{F}_i \\ \mathbf{X}\mathbf{F}_i \\ -\mathbf{I}_I \\ \mathbf{I}_I \end{bmatrix}, \qquad \mathbf{D} := \begin{bmatrix} [\mathbf{R} \ \ -\mathbf{X}\mathbf{F}_\ell] \\ [-\mathbf{R} \ \ +\mathbf{X}\mathbf{F}_\ell] \\ \mathbf{0}_I \\ \mathbf{0}_I \end{bmatrix} \tag{8b}$$

$$\mathbf{e}^\top := \begin{bmatrix} 0.03 \cdot \mathbf{1}_N^\top & 0.03 \cdot \mathbf{1}_N^\top & \bar{\mathbf{q}}_i^\top & \bar{\mathbf{q}}_i^\top \end{bmatrix}. \tag{8c}$$

IV. Computing the Jacobian Matrix via MPP

Consider solving (1) for a particular $\boldsymbol{\theta}$. The corresponding optimal primal-dual pair $(\mathbf{x}_{\boldsymbol{\theta}}, \boldsymbol{\lambda}_{\boldsymbol{\theta}})$ should satisfy the Karush-Kuhn-Tucker (KKT) optimality conditions [21]. Among the inequality constraints of (1), those satisfied with equality at the optimum are termed *active or binding constraints*. Let $(\tilde{\mathbf{C}}, \tilde{\mathbf{D}}, \tilde{\mathbf{e}}, \tilde{\boldsymbol{\lambda}}_{\boldsymbol{\theta}})$ be the row partitions of $(\mathbf{C}, \mathbf{D}, \mathbf{e}, \boldsymbol{\lambda}_{\boldsymbol{\theta}})$ associated with active constraints. Likewise, let $(\bar{\mathbf{C}}, \bar{\mathbf{D}}, \bar{\mathbf{e}}, \bar{\boldsymbol{\lambda}}_{\boldsymbol{\theta}})$ denote the row partition related to *inactive or non-binding constraints*. Leveraging complementarity slackness, the KKT conditions can be expressed as [21]

$$\mathbf{A}\mathbf{x}_{\boldsymbol{\theta}} + \tilde{\mathbf{C}}^\top \tilde{\boldsymbol{\lambda}}_{\boldsymbol{\theta}} = \mathbf{B}\boldsymbol{\theta} \tag{9a}$$

$$\tilde{\mathbf{C}}\mathbf{x}_{\boldsymbol{\theta}} = \tilde{\mathbf{D}}\boldsymbol{\theta} + \tilde{\mathbf{e}} \tag{9b}$$

$$\bar{\mathbf{C}}\mathbf{x}_{\boldsymbol{\theta}} < \bar{\mathbf{D}}\boldsymbol{\theta} + \bar{\mathbf{e}} \tag{9c}$$

$$\tilde{\boldsymbol{\lambda}}_{\boldsymbol{\theta}} > \mathbf{0} \tag{9d}$$

$$\bar{\boldsymbol{\lambda}}_{\boldsymbol{\theta}} = \mathbf{0}. \tag{9e}$$

For (9d) in particular, it is assumed that binding constraints relate to strictly positive dual variables $\tilde{\boldsymbol{\lambda}}_{\boldsymbol{\theta}} > \mathbf{0}$. The reverse is unlikely to occur when $\boldsymbol{\theta}_s$'s are drawn at random. In the unlikely case some of the entries of $\tilde{\boldsymbol{\lambda}}_{\boldsymbol{\theta}}$ do equal zero, the OPF sensitivity for this specific $\boldsymbol{\theta}$ can be ignored and the DNN is trained only to match the OPF minimizer $\mathbf{x}_{\boldsymbol{\theta}}$.

If there are A active constraints, the equalities of (9a)–(9b) constitute a system of $(I+A)$ linear equations over the $(I+A)$ unknowns $(\mathbf{x}_{\boldsymbol{\theta}}, \tilde{\boldsymbol{\lambda}}_{\boldsymbol{\theta}})$. Since $\mathbf{A} \succ \mathbf{0}$, the Lagrangian optimality condition of (9a) yields

$$\mathbf{x}_{\boldsymbol{\theta}} = \mathbf{A}^{-1}(\mathbf{B}\boldsymbol{\theta} - \tilde{\mathbf{C}}^\top \tilde{\boldsymbol{\lambda}}_{\boldsymbol{\theta}}). \tag{10}$$

Substituting (10) into (9b) provides

$$\mathbf{G}\tilde{\boldsymbol{\lambda}}_{\boldsymbol{\theta}} = (\tilde{\mathbf{C}}\mathbf{A}^{-1}\mathbf{B} - \tilde{\mathbf{D}})\boldsymbol{\theta} - \tilde{\mathbf{e}} \tag{11}$$

where $\mathbf{G} := \tilde{\mathbf{C}}\mathbf{A}^{-1}\tilde{\mathbf{C}}^\top$. If matrix $\tilde{\mathbf{C}}$ has linearly independent rows, then $\mathbf{G} \succ \mathbf{0}$, and (11) has a unique solution. Otherwise, there are infinitely many $\tilde{\boldsymbol{\lambda}}_{\boldsymbol{\theta}}$'s satisfying (11) within a shift invariance on the nullspace $\text{null}(\tilde{\mathbf{C}}^\top) = \text{null}(\mathbf{G})$.

For a general QP, the rows of $\tilde{\mathbf{C}}$ are typically linearly independent, thus satisfying the so termed linear independence constraint qualification (LICQ) [21]. Nonetheless, that is not the case for the inverter dispatch problem of (7) where instances of linearly dependent active constraints occur frequently. To see this, consider the feeder of Fig. 1, where buses $\{1,2\}$ host inverters, bus 3 is a load bus, and 4 is a zero-injection bus. Let x_n denote the reactance for the line feeding bus n. Then, matrix $\mathbf{X}$ can be found as [17]

$$\mathbf{X} = \begin{bmatrix} x_1 & x_1 & x_1 & x_1 \\ x_1 & x_1+x_2 & x_1 & x_1 \\ x_1 & x_1 & x_1+x_3 & x_1+x_3 \\ x_1 & x_1 & x_1+x_3 & x_1+x_3+x_4 \end{bmatrix}.$$

Moreover, the product $\mathbf{X}\mathbf{F}_i$ selects the first two columns of $\mathbf{X}$ corresponding to inverter locations. Now consider a loading scenario that causes the voltage at bus 3 to be at the lower limit. Being a zero-injection bus, the downstream

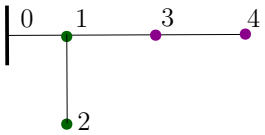

Fig. 1. A 5-bus feeder demonstrating the violation of LICQ for (7).

bus 4 shares the same voltage. Thus, the minimum voltage constraints become active for buses 3 and 4. Assuming no other constraints being active, matrix $\tilde{\mathbf{C}}$ of (8b) reads as

$$\tilde{\mathbf{C}} = -\begin{bmatrix} x_1 & x_1 \\ x_1 & x_1 \end{bmatrix}.$$

which is clearly row rank-deficient. This counterexample emphasizes the need to explicitly address the case where the solution of (1) or (7) fails to meet the LICQ.

Regardless of whether $\tilde{\mathbf{C}}$ is full row-rank or not, the solution to (11) lies in the polyhedron

$$\tilde{\mathbf{\Lambda}}_{\boldsymbol{\theta}} := \{\tilde{\boldsymbol{\lambda}}_{\boldsymbol{\theta}} = \mathbf{G}^{+}(\tilde{\mathbf{C}}\mathbf{A}^{-1}\mathbf{B} - \tilde{\mathbf{D}})\boldsymbol{\theta} - \mathbf{G}^{+}\tilde{\mathbf{e}} + \mathbf{u}, \tilde{\mathbf{C}}^{\top}\mathbf{u} = \mathbf{0}\}$$

where $\mathbf{G}^{+}$ is the pseudo-inverse of $\mathbf{G} = \tilde{\mathbf{C}}\mathbf{A}^{-1}\tilde{\mathbf{C}}^{\top}$. Substituting $\tilde{\boldsymbol{\lambda}}_{\boldsymbol{\theta}}$ back in (10) and exploiting $\tilde{\mathbf{C}}^{\top}\mathbf{u} = \mathbf{0}$, we get

$$\mathbf{x}_{\boldsymbol{\theta}} = \mathbf{J}_{\boldsymbol{\theta}}\boldsymbol{\theta} + \mathbf{A}^{-1}\tilde{\mathbf{C}}^{\top}\mathbf{G}^{+}\tilde{\mathbf{e}}. \tag{12}$$

where the sensitivity matrix is defined as

$$\mathbf{J}_{\boldsymbol{\theta}} := \mathbf{A}^{-1}\mathbf{B} - \mathbf{A}^{-1}\tilde{\mathbf{C}}^{\top}\mathbf{G}^{+}(\tilde{\mathbf{C}}\mathbf{A}^{-1}\mathbf{B} - \tilde{\mathbf{D}}). \tag{13}$$

Although there may be infinitely many $\tilde{\boldsymbol{\lambda}}_{\boldsymbol{\theta}}$'s satisfying the KKT conditions, the optimal primal solution of (1) is unique if it exists. This is not surprising since (1) has a strictly convex objective. Moreover, the solution can be expressed as an affine function of $\boldsymbol{\theta}$. Note that the parameters of this affine function depend on the set of active constraints, and this is indicated by the subscript of $\boldsymbol{\theta}$ on $\mathbf{J}_{\boldsymbol{\theta}}$.

Because the triplet $(\boldsymbol{\theta}, \mathbf{x}_{\boldsymbol{\theta}}, \tilde{\boldsymbol{\lambda}}_{\boldsymbol{\theta}})$ should also satisfy (9d) and (9c), there exists a $\mathbf{u} \in \text{null}(\tilde{\mathbf{C}}^{\top})$ so that $\boldsymbol{\theta}$ satisfies

$$\mathbf{G}^{+}(\tilde{\mathbf{C}}\mathbf{A}^{-1}\mathbf{B} - \mathbf{D})\boldsymbol{\theta} > \mathbf{G}^{+}\tilde{\mathbf{e}} - \mathbf{u} \tag{14a}$$

$$(\bar{\mathbf{C}}\mathbf{J}_{\boldsymbol{\theta}} - \bar{\mathbf{D}})\boldsymbol{\theta} < \bar{\mathbf{e}} - \bar{\mathbf{C}}\mathbf{A}^{-1}\tilde{\mathbf{C}}^{\top}\mathbf{G}^{+}\tilde{\mathbf{e}} \tag{14b}$$

So far, we have characterized the solution to (1) for a particular $\boldsymbol{\theta}$. Since we are interested in the sensitivity of the OPF minimizer with respect to $\boldsymbol{\theta}$, we ask the natural question whether (12) holds for all $\boldsymbol{\theta}'$'s within a vicinity of this specific $\boldsymbol{\theta}$. The answer to this question is on the affirmative. To see this, fix $\mathbf{u}$ and perturb $\boldsymbol{\theta}$ to get $\boldsymbol{\theta}'$ so it still satisfies (14). Using $\boldsymbol{\theta}'$ in lieu of $\boldsymbol{\theta}$, construct $\mathbf{x}_{\boldsymbol{\theta}'}$ from (12), and $\tilde{\boldsymbol{\lambda}}_{\boldsymbol{\theta}'}$ from (11). In doing so, we row-partition matrices assuming the same constraints are active as for $\boldsymbol{\theta}$. This means we still use matrix $\mathbf{J}_{\boldsymbol{\theta}}$. We also set $\bar{\boldsymbol{\lambda}}_{\boldsymbol{\theta}'} = \mathbf{0}$. The constructed primal-dual pair $(\mathbf{x}_{\boldsymbol{\theta}'}, \boldsymbol{\lambda}_{\boldsymbol{\theta}'})$ satisfies the KKT conditions of (1) for $\boldsymbol{\theta}'$, and hence constitutes an optimal solution for this $\boldsymbol{\theta}'$. The aforesaid process can be repeated for any $\boldsymbol{\theta}'$ in the vicinity of the original $\boldsymbol{\theta}$ because (14) are strict inequalities. In other words, formula (12) is valid for all $\boldsymbol{\theta}'$'s around $\boldsymbol{\theta}$ and with matrix $\mathbf{J}_{\boldsymbol{\theta}}$ remaining unaltered.

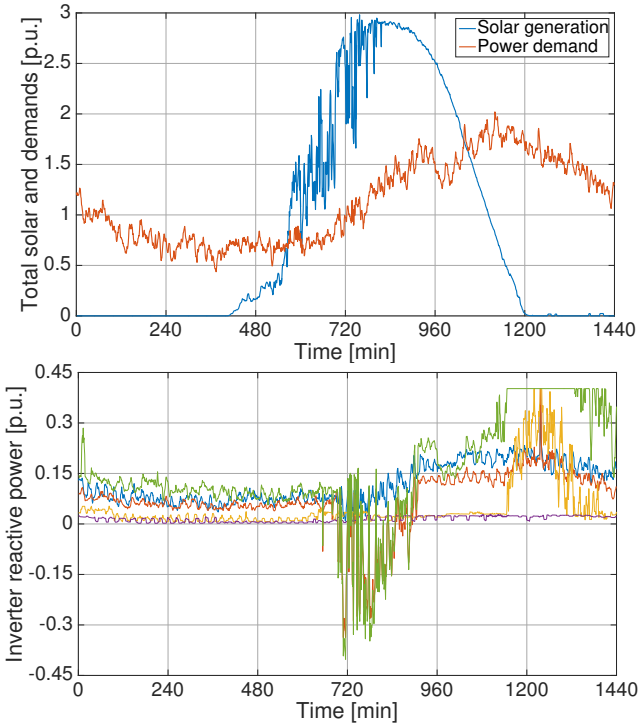

Fig. 2. Minute-based data for the total (feeder-wise) solar power generation and active power demand (top panel); and the optimal reactive power injection setpoints for five inverters (bottom panel).

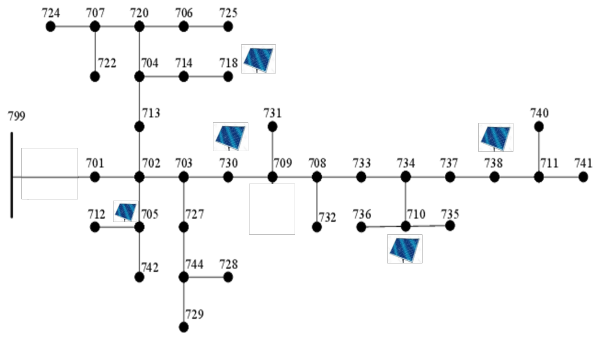

Fig. 3. A modified IEEE 37-bus feeder showing additional solar generators.

The latter reveals that the OPF operator from $\boldsymbol{\theta}$ to $\mathbf{x}_{\boldsymbol{\theta}}$ is differentiable around $\boldsymbol{\theta}$. In addition, its Jacobian matrix is provided in closed form using (13). Calculating $\mathbf{J}_{\boldsymbol{\theta}}$ entails: *i)* Knowing the set of active constraints; *ii)* inverting matrix $\mathbf{A}$ once; and *iii)* inverting matrix $\mathbf{G}$. Although *iii)* is executed once per $\boldsymbol{\theta}$, the computation is lightweight since the number of active constraints A should be smaller than N. In a nutshell, computing $\mathbf{J}_{\boldsymbol{\theta}}$ once (1) has been solved for a particular $\boldsymbol{\theta}$, is much simpler than solving (1) per se. Hence, computing sensitivities adds insignificant complexity in the process of constructing the labeled dataset $(\boldsymbol{\theta}, \mathbf{x}_{\boldsymbol{\theta}}, \mathbf{J}_{\boldsymbol{\theta}})$.

V. Numerical Tests

The performance of the developed approach was numerically tested by solving the OPF in (7) for the IEEE 37-

bus feeder upon removing regulators, incorporating 5 solar generators, and converting it to its single-phase equivalent; see Fig. 3. We extracted minute-based load and solar generation data for June 1, 2018, from the Pecan Street dataset [22]. The feeder has 25 buses with non-zero load. The first 75 non-zero load buses from the dataset were aggregated every 3 and normalized to obtain 25 load profiles. Similarly, 5 solar generation profiles were obtained. The normalized load profiles for the 24-hr period were scaled so the 97-th percentile of the total load duration curve coincided with the total nominal load. This scaling results in a peak aggregate load being 1.1 times the total nominal load. We synthesized reactive loads by scaling active demand to match the power factors of the IEEE 37-bus feeder. The 5 solar generators were of equal ratings and scaled to meet 75% of the total energy requirement over the entire day. Figure 2 shows the total demand and solar generation across the feeder (top), along with the optimal reactive power injection for each of the five inverters (bottom). The optimal setpoints were obtained by solving (7) using YALMIP and Sedumi. In solving the 1,440 OPF instances, no constraints were active for 914 instances. Out of the remaining 526 instances, the LICQ was not satisfied for 175 instances; thus necessitating the approach pursued here.

Our tests compared the P-DNN and SI-DNN, both trained to predict the minimizer of (7). For the first set of tests, the DNNs were assumed to be trained on an hourly basis. To evaluate the potential benefit of integrating sensitivity information, the architecture, optimizer, and training epochs were kept identical for the two DNNs. In fact, the architecture was chosen in favor of the P-DNN. It was determined so that for a sufficiently large training set, the P-DNN would be able to predict a near-optimal inverter dispatch. This ensured a sufficiently rich, yet not too complex architecture: Three hidden layers with 210, 210, and 350 neurons respectively, all using the rectified linear unit (ReLU) activation. All results were generated using the Adam optimizer with a learning rate of 0.01. The DNNs were implemented using the TensorFlow library on Google colab.

For a given hour, the two DNNs were trained using 4 OPF instances, obtained by sampling the upcoming hourly period every 15 min. The remaining 56 grid instances were used to test the two DNNs based on the mean square error (MSE) $\|\mathbf{x}_s - \hat{\mathbf{x}}_s\|_2^2$ on the predicted setpoints. To compare P-DNN and SI-DNN under varying conditions, their training and testing errors were evaluated over hours 5, 12, and 20, and shown in Fig. 4. The tests show that even when P-DNN yields smaller training errors, SI-DNN offers an improvement in testing error by one or two orders of magnitude.

We next conducted the previous test but now over 10 Monte Carlo runs, each time selecting the training OPF scenarios at random. To evaluate the effect of the dataset size, we trained both DNNs using 4 and 12 scenarios for each of the hours 5, 12, 16, and 20. Table I reports the test MSEs attained after 1,000 epochs and averaged over the 10 Monte Carlo runs. The results corroborates that the SI-DNN features a gain in prediction accuracy over P-DNN by 1-3 orders of magnitude. Its advantage is generally more pronounced when dealing with smaller datasets, as expected.

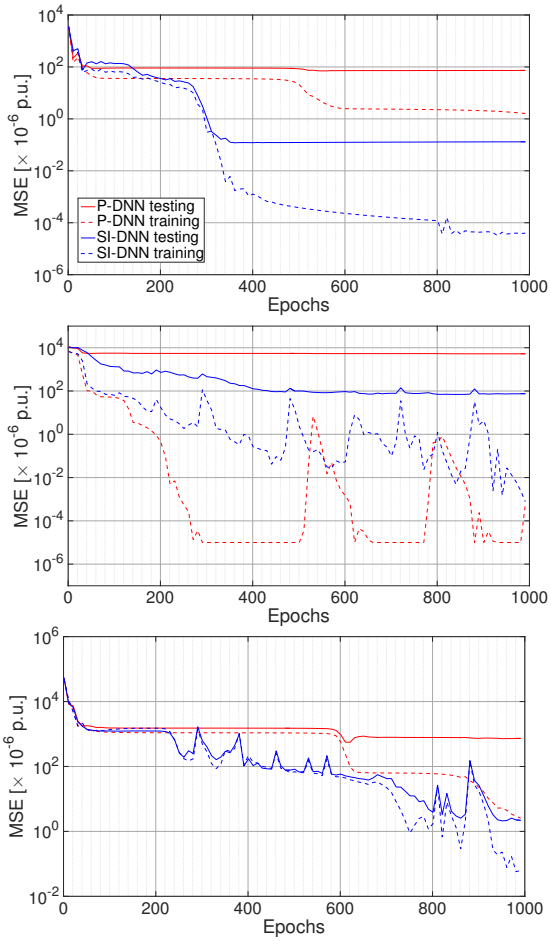

Fig. 4. Training and testing errors across epochs in terms of MSE for hours 5 (top), 12 (middle), and 20 (bottom). Training was performed using 15-min smart meter data (4 OPF instances) to operate during the hour of interest.

TABLE I
AVERAGE TEST MSE [IN 10^{-6} PU] AFTER 1,000 EPOCHS FOR DIFFERENT HOURS OF THE DAY

Hour	12 OPF training scenarios		4 OPF training scenarios	
	P-DNN	**SI-DNN**	**P-DNN**	**SI-DNN**
5	27.9	0.04	75.3	0.33
12	1184.4	50.4	7229.8	4755.3
16	145.1	49.6	540.3	342.5
20	359.3	45.0	1500.6	145.2

For the last set of tests, the two DNNs were trained to learn the OPF for an entire day. The training dataset was constructed by sampling 5%, 10%, and 20% of the one-minute data. To explicitly focus on periods of high variability, we excluded the hours from midnight to 10 am from sampling. Table II summarizes the MSEs obtained after 1,000 epochs and averaged over 100 Monte Carlo draws of the training dataset. The table also reports the average training time for 1,000 epochs for the two methodologies. The runtimes on Google colab often varies with session. Thus, these times can only be compared separately for each training scenario; and not across scenarios. The results establish that the SI-

TABLE II
AVERAGE TEST MSE [IN 10^{-6} PU] AND TRAINING TIME [IN SEC] AFTER 1,000 EPOCHS FOR 10–12 AM (840 MIN-BASED SCENARIOS)

Training Scenarios	P-DNN		SI-DNN	
	MSE	**Time**	**MSE**	**Time**
5%	1407.8	108.9	119.9	118.7
10%	520.2	77.1	72.9	79.7
20%	219.2	100.8	55.1	113.4

DNN consistently outperforms the P-DNN without incurring any significant increase in training time. For example, the SI-DNN achieves an MSE of $119.9 \cdot 10^{-6}$ pu using 5% of the data, whereas the P-DNN achieves an MSE of $219.2 \cdot 10^{-6}$ pu although it has been trained by using 4 times more data (20%).

VI. CONCLUSIONS AND ONGOING WORK

A novel procedure for training DNNs that learn to optimize has been put forth. Leaping beyond the general practice of training DNNs via labeled parameter-minimizer pairs, this work ensures that the sensitivities of DNN predictions to inputs are close to the respective sensitivities of the original optimization task. Addressing QPs in particular, the sensitivities required for training the DNN can be computed readily in virtue of results from MPP theory. The application pursued has been inverter reactive power control in distribution systems for minimizing losses subject to voltage constraints. It has been shown here that although for general QPs dual degeneracies are rare, for the inverter dispatch task such degeneracies can be encountered frequently. Fortunately, even for such instances, the required Jacobian matrices do exist in general, and the proposed approach can successfully compute them in closed form. The efficacy of the novel training method is demonstrated via numerical tests that corroborate an improvement in prediction accuracy by 2-3 orders of magnitude, as compared to the traditional regression approach. The improvements are more pronounced in the small-data regime, where a DNN has to learn to optimize using few examples.

Prompted by these promising results, we are currently generalizing this creative idea of including gradient information into DNNs towards several exciting directions. Beyond MPQPs that feature rich structure in their solutions, we are interested in improving learning for resource allocation tasks of the generic parametric form

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

Thanks to results from optimization, given $(f, \mathbf{g}, \mathbf{x}_{\boldsymbol{\theta}}, \boldsymbol{\lambda}_{\boldsymbol{\theta}})$ one can compute the gradient $\nabla_{\boldsymbol{\theta}} f(\mathbf{x}_{\boldsymbol{\theta}})$, and the Jacobian matrices $\nabla_{\boldsymbol{\theta}} \mathbf{x}_{\boldsymbol{\theta}}$ and $\nabla_{\boldsymbol{\theta}} \boldsymbol{\lambda}_{\boldsymbol{\theta}}$, regardless if $(P_{\boldsymbol{\theta}})$ is convex. Leveraging these results, we are currently working towards: *d1)* incorporating sensitivities for learning to optimize different power system optimization and monitoring tasks, including the AC OPF and its various convex relaxations; *d2)* integrating dual sensitivities and weight their deviations; and *d3)* predicting binding constraints so that exact minimizers can be found by solving an OPF over a condensed feasible set.

REFERENCES

[1] M. Eisen, C. Zhang, L. F. O. Chamon, D. D. Lee, and A. Ribeiro, "Learning optimal resource allocations in wireless systems," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2775–2790, May 2019.

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

[3] 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.

[4] 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 Process.*, vol. 66, no. 20, pp. 5438–5453, Oct. 2018.

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

[6] 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.

[7] A. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," 2019, preprint. [Online]. Available: https://arxiv.org/abs/1910.01213

[8] 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

[9] 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.

[10] Y. Chen, Y. Shi, and B. Zhang, "Input convex neural networks for optimal voltage regulation," 2020, (submitted). [Online]. Available: https://arxiv.org/abs/2002.08684

[11] Q. Zhang, K. Dehghanpour, Z. Wang, F. Qiu, and D. Zhao, "Multi-agent safe policy learning for power management of networked microgrids," 2020, (submitted). [Online]. Available: https://arxiv.org/abs/1907.02091v3

[12] W. Wang, N. Yu, Y. Gao, and J. Shi, "Safe off-policy deep reinforcement learning algorithm for Volt-VAR control in power distribution systems," *IEEE Trans. Smart Grid*, pp. 1–1, 2019.

[13] 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.

[14] 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.

[15] Q. Zhou, L. Tesfatsion, and C.-C. Liu, "Short-term congestion forecasting in wholesale power markets," *IEEE Trans. Power Syst.*, vol. 26, no. 4, pp. 2185–2196, Nov. 2011.

[16] Y. Ji, R. J. Thomas, and L. Tong, "Probabilistic forecasting of real-time LMP and network congestion," *IEEE Trans. Power Syst.*, vol. 32, no. 2, pp. 831–841, Mar. 2017.

[17] S. Taheri, M. Jalali, V. Kekatos, and L. Tong, "Fast probabilistic hosting capacity analysis for active distribution systems," *IEEE Trans. Smart Grid*, Feb. 2020, (submitted). [Online]. Available: https://arxiv.org/abs/2002.01980

[18] S. Taheri, V. Kekatos, and H. Veeramachaneni, "Strategic investment in energy markets: A multiparametric programming approach," *IEEE Trans. Smart Grid*, Apr. 2020, (submitted). [Online]. Available: https://arxiv.org/abs/2004.06483

[19] 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.

[20] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht, "Smart*: An open data set and tools for enabling research in sustainable homes," in *Proc. Workshop on Data Mining Applications in Sustainability*, Beijing, China, Aug. 2012.

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

[22] (2018) Dataport. Pecan Street Inc. [Online]. Available: https://dataport.cloud/