Controlling Smart Inverters using Proxies: A Chance-Constrained DNN-based Approach

Sarthak Gupta, Student Member, IEEE, Vassilis Kekatos, Senior Member, IEEE and Ming Jin, Member, IEEE

Abstract—Coordinating inverters at scale under uncertainty is the desideratum for integrating renewables in distribution grids. Unless load demands and solar generation are telemetered frequently, controlling inverters given approximate grid conditions or proxies thereof becomes a key specification. Although deep neural networks (DNNs) can learn optimal inverter schedules, guaranteeing feasibility is largely elusive. Rather than training DNNs to imitate already computed optimal power flow (OPF) solutions, this work integrates DNN-based inverter policies into the OPF. The proposed DNNs are trained through two OPF alternatives that confine voltage deviations on the average and as a convex restriction of chance constraints. The trained DNNs can be driven by partial, noisy, or proxy descriptors of the current grid conditions. This is important when OPF has to be solved for an unobservable feeder. DNN weights are trained via back-propagation and upon differentiating the AC power flow equations assuming the network model is known. Otherwise, a gradient-free variant is put forth. The latter is relevant when inverters are controlled by an aggregator having access only to a power flow solver or a digital twin of the feeder. Numerical tests compare the DNN-based inverter control schemes with the optimal inverter setpoints in terms of optimality and feasibility.

Index Terms—Stochastic gradient descent; deep neural networks; primal-dual updates; inverter control; inverse function theorem; reactive power compensation; stochastic optimization; chance constraints.

I. INTRODUCTION

The high penetration of DERs (such as rooftop photovoltaics, batteries, and demand response devices) introduces additional variability in distribution grid operation. Uncontrolled variations in power injections can in turn induce abrupt fluctuations in nodal voltages. Fortunately, the smart inverters interfacing DERs with the grid can propel their integration by additionally providing reactive power support. The coordinated control of DERs across a feeder can be formulated as an OPF [1], [2]. If the grid is modelled using the AC power flow equations, tackling this OPF using conventional optimizationbased solvers becomes formidable as DERs increase in numbers and need to be redispatched frequently under dynamic conditions. At the same time, solving an OPF presumes that all problem inputs (load demands and solar generation) are precisely known. Nonetheless, such parameters are oftentimes described stochastically, observed under noise and delays, or the operator can monitor only few of them in real time.

Alternatively, recent literature advocates the use of machine learning (ML) models to solve minimization problems under the *learning-to-optimize* paradigm. Due to the rich modeling and fast inference capabilities of ML models, learning-tooptimize becomes relevant to scenarios where large-scale non-linear optimization tasks have to be solved frequently and/or under uncertain or partially observed inputs. MLbased schemes for tackling with the OPF have already been explored and can be broadly classified into the OPF-then-learn and OPF-and-learn categories. Methods within the former category involve two steps. They first generate a labelled training dataset by solving a large number of OPFs. The features and labels of this dataset consist of the OPF input parameters and outputs (optimal DER setpoints), respectively. During the second step, an ML model is trained to predict the dataset labels in the conventional supervised manner. Within the OPF-then-learn category, kernel-based regression has been employed to learn inverter control rules in [3], while DNNs have been trained to predict OPF solutions under a linearized [4] and the exact AC grid models [5], [6], [7], [8]. To expedite the first step, the sensitivity-informed learning method of [9] and [10] trains a DNN to match not only the OPF minimizers, but also their partial derivatives with respect to the OPF inputs. Despite the data efficiency enhancement offered by sensitivity-informed learning, the OPF-then-learn strategy lacks feasibility guarantees and presumes OPF input parameters are deterministic and known. Furthermore, generating labels by solving several OPFs incurs a significant computational overhead. Consequently, the OPF-then-learn strategy is not well suited for scenarios where the optimal policies need to be re-learned frequently on account of changing underlying data distribution.

1

Instead of following the two-step approach of first generating OPF labels and then training the ML model, OPF-and*learn* approaches learn the ML model directly while solving the OPF in a single step. Hence, methods within the OPF-and*learn* category are more suited for dynamic applications where ML models needs to be continuously retrained. Under the OPF-and-learn paradigm, reference [11] adopts support vector machines (SVMs) to design inverter control rules, adjusted to grid conditions in a quasi-stationary fashion. Albeit SVMbased rules can be learned via a convex program, kernel functions have to be specified beforehand. In [12], inverter control rules are optimized along with capacitor status decisions to minimize voltage deviations using a two-timescale reinforcement learning (RL) approach yet no feeder-level constraints are involved. Enforcing network constraints is challenging for ML-based OPF methods. One could heuristically project the ML prediction for the OPF solution [5], [11], or consider penalizing constraint deviations [3], [4], [12].

An alternative for dealing with constraints in the learning-

This work was supported by the U.S. National Science Foundation under grants 1751085 and 2034137.

to-optimize process is through the discounted return functions used in RL approaches. In this context, reference [13] updates DNN-based inverter policies continuously by successively linearizing feeder constraints. A similar *safe RL* learning scheme is put forth in [14], which focuses on regulating the number of voltage violations across nodes through a method of multipliers strategy. However, both these works updated the policy parameters by solving an optimization problem at every update step that might be computationally intensive, restricting their applicability on dynamic scenarios. Secondly, both [13] and [14] focus on scenarios where measurements across feeder nodes are available to policies in real time. Finally, RL-based approaches are in general more complex in implementation, which can be hindering their adoption by grid operators; e.g., the *safe RL* strategy of [14] involves nine different DNNs.

Primal-dual learning [15], provides a computationally less intensive alternative to RL for handling feeder constraints. Different from [13] and [14], primal-dual learning involves simpler gradient-based updates of the policy parameters and the related dual variables alike. Stochastic primal-dual updates were first applied towards learning the optimal control policies for smart inverters to enforce averaged voltage constraints in conference precursor [16] of our work.

This work extends [16] in four fronts. Firstly, the underlying feeder is represented using the exact AC model rather than the approximate linearized model [17] previously employed. This extension is non-trivial as the gradients needed for the stochastic primal-dual updates of the DNN are now found in an indirect manner using the underlying AC power flow equations and the inverse function theorem. Secondly, we illustrate the versatility of stochastic primal-dual updates as they can cope with probabilistic voltage constraints via convex restrictions [18]. Thirdly, we demonstrate that primal-dual learning can also be employed when only partial information is available in real-time. This adheres to practical setups where the utility might have real-time telemetry only over a subset of grid locations. Fourthly, we propose gradient-free counterparts of the primal-dual updates that train the DNN knowing only the values of voltages and injections. Such approaches are useful when: i) DERs are optimized by an entity other than the utility (e.g., an aggregator) that does not know the feeder model and can only access a digital twin of the grid; *ii*) The feeder model is known but complex (due to the presence of transformers, capacitors, ZIP loads) and differentiation becomes perplex, but the utility has access to a power flow solver; and *iii*) The feeder model is incomplete (secondary networks or regulator taps are unknown), but the utility has access to smart meter readings.

The rest of the paper is organized as follows. Section II formulates the task of DNN-based smart inverter control, and puts forth an *averaged* and a *probabilistic* scheme. Section III elaborates on primal-dual DNN learning for both schemes. Section III calculates the gradients needed for the stochastic updates, while the gradient-free implementation is presented in Section IV. The novel DNN-based inverter control strategies are evaluated using real-world data on the IEEE 37-bus feeder in Section V. Conclusions are drawn in Section VI.

Notation: Lower- (upper-) case boldface letters denote col-

umn vectors (matrices), and calligraphic symbols are reserved for sets. Symbol $^{\top}$ stands for transposition and $||\mathbf{x}||_2$ denotes the ℓ_2 -norm of \mathbf{x} . Vectors $\mathbf{0}$ and $\mathbf{1}$ are respectively the vectors of all zeros and ones of appropriate dimensions.

II. PROBLEM FORMULATION

Consider a feeder with N + 1 buses. The substation is indexed by 0 and the remaining buses comprise the set $\mathcal{N} := \{1, \dots, N\}.$

Let $p_n + jq_n$ be the complex power injection at bus *n*. Its active power component can be decomposed as $p_n = p_n^g - p_n^c$, where p_n^g is the solar generation and p_n^c the inelastic load at bus *n*. Its reactive power component can be similarly expressed as $q_n = q_n^g - q_n^c$. The vectors (\mathbf{p}, \mathbf{q}) collecting the power injections at all non-substation buses can be decomposed as $\mathbf{p} = \mathbf{p}^g - \mathbf{p}^c$ and $\mathbf{q} = \mathbf{q}^g - \mathbf{q}^c$.

Let vector parameter $\theta := \{\mathbf{p}^c, \mathbf{q}^c, \mathbf{p}^g\} \subseteq \mathbb{R}^{3N}$ collect the loads (active and reactive) and active solar generation at all non-substation buses. We will henceforth term θ as the vector of *grid conditions*. Given θ , the task of reactive power control by DERs aims at optimally setting \mathbf{q}^g to minimize a feederwide objective while complying with network and inverter limitations. Starting with the latter, the reactive power injected by inverter *n* is limited by its given apparent power limit \bar{s}_n . Apparent power constraints are local and will be collectively denoted by

$$\mathbf{q}^{g} \in \mathcal{Q}_{\boldsymbol{\theta}} := \left\{ \mathbf{q}^{g} : |q_{n}^{g}| \leq \sqrt{\overline{s}_{n}^{2} - (p_{n}^{g})^{2}} \,\,\forall n \right\}.$$
(1)

where the subscript in Q_{θ} denotes that the feasible space changes as the value of solar generation \mathbf{p}_q changes.

The task of coordinating inverters can be centrally handled by an aggregator assigned to manage these DERs. The aggregator finds the reactive power setpoints for DERs by minimizing ohmic losses on distribution lines while maintaining voltage magnitudes within per-bus bounds $[\mathbf{v}, \overline{\mathbf{v}}]$ as

$$\min_{\mathbf{q}_t \in \mathcal{Q}_{\boldsymbol{\theta}}} \quad \ell(\mathbf{q}, \boldsymbol{\theta}) \tag{2}$$

s.to $\mathbf{v} < \mathbf{v}(\mathbf{q}, \boldsymbol{\theta}) < \overline{\mathbf{v}}.$

where **v** is the vector of bus voltage magnitudes. We will henceforth refer to voltage magnitudes as voltages unless stated otherwise. We slightly abuse notation and use **q** in lieu of \mathbf{q}^g to unclutter notation, as \mathbf{q}^c is included in θ anyway. Note that expressions $\ell(\mathbf{q}, \theta)$ and $\mathbf{v}(\mathbf{q}, \theta)$ capture the dependence of losses and nodal voltages on the reactive setpoints of DERs **q** under the current grid conditions θ .

Solving (2) can be computationally and communicationwise taxing if θ changes frequently. Moreover, by the time (2) is solved and optimal setpoints are downloaded to DERs, grid conditions θ may have changed rendering the computed setpoints obsolete. To account for the uncertainty in θ , we propose two stochastic formulations. The first formulation replaces $\ell(\mathbf{q}, \theta)$ and $\mathbf{v}(\mathbf{q}, \theta)$ with their averages:

$$\min_{\mathbf{q}_{\ell} \in \mathcal{Q}_{\boldsymbol{\theta}}} \mathbb{E}[\ell(\mathbf{q}, \boldsymbol{\theta})]$$
s.to $\mathbf{v} < \mathbb{E}[\mathbf{v}(\mathbf{q}, \boldsymbol{\theta})] < \overline{\mathbf{v}}$
(3)

where the expectation \mathbb{E} is with respect to θ . We refer to (3) as the *averaged formulation*. While the averaged formulation takes care of uncertainties in θ , the obtained setpoints may violate the voltage limits in (2) quite frequently. This undesirable behavior results from the fact constraining the average value of voltages does not provide strong guarantees on their perinstance values. Nevertheless, the averaged formulation has an attractive structure that permits straightforward stochastic gradient descent (SGD)-based steps to arrive at the optimal setpoints as we will see later.

A more conservative approach is possible through the *probabilistic formulation*

$$\min_{\mathbf{q}\in\mathcal{Q}_{\boldsymbol{\theta}}} \mathbb{E}[\ell(\mathbf{q},\boldsymbol{\theta})] \tag{4a}$$

s.to
$$\Pr[\underline{v}_n \ge v_n(\mathbf{q}, \boldsymbol{\theta})] \le \alpha, \ \forall n \in \mathcal{N}$$
 (4b)

$$\Pr\left[\overline{v}_n \le v_n(\mathbf{q}, \boldsymbol{\theta})\right] \le \alpha, \ \forall n \in \mathcal{N}$$
(4c)

where (4b)–(4c) ensure each bus voltage remains within the desired limits with a probability of at least $1 - \alpha$ on each side. Here $\alpha \in (0, 1)$ is a small *violation probability*. In contrast to (3), the formulation in (4) focuses on restricting the frequency of occurrence of voltage violations. Problem (4) can be rewritten as

$$\min_{\mathbf{q}\in\mathcal{Q}_{\boldsymbol{\theta}}} \mathbb{E}[\ell(\mathbf{q},\boldsymbol{\theta})]$$
(5a)

s.to
$$\mathbb{E}\left[\mathbb{1}(\underline{v}_n - v_n(\mathbf{q}, \boldsymbol{\theta}))\right] \le \alpha, \ \forall n \in \mathcal{N}$$
 (5b)

$$\mathbb{E}\left[\mathbb{1}(v_n(\mathbf{q}, \boldsymbol{\theta}) - \overline{v}_n)\right] \le \alpha, \ \forall n \in \mathcal{N}$$
 (5c)

where the indicator function $\mathbb{1}(x)$ is defined as

$$\mathbb{1}(x) = \begin{cases} 1 & , \ x \ge 0 \\ 0 & , \ x < 0 \end{cases}$$
(6)

The probabilistic formulation is difficult to handle since the indicator function is neither convex nor differentiable. In quest of workable alternatives, Section III pursues convex approximations of constraints (5b)–(5c).

For now, let both formulations be represented by the general stochastic program

$$\min_{\mathbf{q}\in\mathcal{Q}_{\boldsymbol{\theta}}} \mathbb{E}[\ell(\mathbf{q},\boldsymbol{\theta})]$$
(7)
s.to $\mathbb{E}[\mathbf{g}(\mathbf{q},\boldsymbol{\theta})] \leq \mathbf{0}.$

Note that solving (7) results in a single 'one-size-fits-all' **q** that does not adapt to different θ 's. To render DER setpoints responsive to grid conditions, we resort to a *control policy*, where the reactive setpoints **q** are captured by a function $\pi(\theta; \mathbf{w})$, which is parameterized by **w**.

Ideally, the control policy is driven by the vector of grid conditions θ . Nevertheless, during real-time operation, the operator controlling the DERs may not be able to observe the complete θ . Instead, it may have to act upon a proxy ϕ of the actual θ . The DER control policy driven by ϕ can then be found by solving the constrained stochastic minimization

$$\min_{\mathbf{w}: \boldsymbol{\pi}(\boldsymbol{\phi}; \mathbf{w}) \in \mathcal{Q}_{\boldsymbol{\theta}}} \mathbb{E}[\ell(\boldsymbol{\pi}(\boldsymbol{\phi}; \mathbf{w}), \boldsymbol{\theta})]$$
s.to $\mathbb{E}[\mathbf{g}(\boldsymbol{\pi}(\boldsymbol{\phi}; \mathbf{w}), \boldsymbol{\theta})] \le \mathbf{0}.$
(8)

The DER control policies found through (8) are adaptive to the proxy vector ϕ and the optimization is over the



Fig. 1. Real-time operation of the proposed DER inverter control scheme. Proxy vector ϕ consisting of measurements from nodes $\{2, 3, 4, 8\}$ is transmitted to the aggregator. The DNN-based policy acts upon ϕ to predict setpoints **q**, which are then broadcast to the inverters at buses 2 and 8.

parameters w. Policies account for the uncertainty over θ , and correspondingly ϕ . Note that the expectations in (8) couple the system's performance across OPF instances of θ .

The proxy vector ϕ can be chosen to represent the operational setup for which the control policies are being designed. In the absence of real-time measurements from all nodes, and/or to save on communication overhead, vector ϕ can consist of active line flows from distribution lines [16]. Meteorological data such as solar irradiance and ambient temperature, which serve as surrogates for **p**, can also be included in ϕ . One can also explore convolutional neural networks (CNNs)-based policies that accept sky images in place of solar irradiance measurements as inputs to be included in ϕ . In Section V, a more straightforward scenario is explored whereby measurements from a subset of buses in \mathcal{N} are assumed to be available in real-time, resulting in $\phi \subset \theta$.

Previous works have studied linear inverter control policies of the form $\pi(\phi; \mathbf{w}) = \mathbf{w}^{\top} \phi$; see e.g., [19], [20], [21]. Nonetheless, the optimal policy $\pi(\phi; \mathbf{w})$ is not necessarily affine in ϕ , especially when ϕ is a proxy for θ . The grand challenge towards scalable inverter control is to design *nonlinear control policies*. To this end, in [11], we modeled $\pi(\phi; \mathbf{w})$ as a support vector machine (SVM) and designed the policy through an OPF formulation. The advantage of SVM-based policies is that they can be trained to optimality using convex optimization. Nonetheless, selecting the appropriate kernel and control input ϕ can be challenging. Inspired by their fieldchanging performance in various engineering tasks, here we propose capturing the DER control policy $\pi(\phi; \mathbf{w})$ using DNNs, and learn weights \mathbf{w} in a data-driven physics-aware fashion.

Figure 1 depicts the real-time operation of the proposed control strategy. The smart inverters to be controlled are located on buses 2 and 8. The proxy vector $\phi \subset \theta$, consisting of $\{p_n^c, q_n^c, p_n^g\}$ measurements from buses 2, 3, 4 and 8, is transmitted to the operator. The operator feeds ϕ as an input to the DNN-based policy and predicts setpoints **q**. These setpoints are then broadcast to DER inverters for implementation. It is worth emphasizing here that although the operator needs



Fig. 2. DNN-based policy with modified activation layer to ensure $\pi(\phi; \mathbf{w}) \in \mathcal{Q}_{\theta}$ for all ϕ . The output neuron feeds into the tanh() activation function and then scaled by $\sqrt{\overline{s}_n^2 - (p_n^g)^2}$. The scaling operation involves a skip connection from p_n^g , which is one of the inputs to the DNN.

to know pairs of (θ, ϕ) during training, the DNN-based policy operates acting solely on ϕ .

III. PRIMAL-DUAL DNN TRAINING

The stochastic formulation in (8) is challenging to solve on account of the expectation operator in both the objective and constraints. Computing the needed expectations requires knowing the probability density functions (pdf) of ϕ and θ . Even if these pdfs are known, computing the expectations is still non-trivial granted the policies $\pi(\phi; \mathbf{w})$ are non-linear in ϕ . These complications promulgate a stochastic approximation approach towards solving (8). In the conventional machine learning setup, the weights of a DNN are found by minimizing a data-fitting loss function under no constraints via stochastic gradient descent. Here, to accommodate constraints, we adopt the stochastic primal-dual updates of [15] as presented next.

Consider the Lagrangian function of the problem in (8)

$$L(\mathbf{w};\boldsymbol{\lambda}) := \mathbb{E}[\ell(\boldsymbol{\pi}(\boldsymbol{\phi};\mathbf{w}),\boldsymbol{\theta})] + \boldsymbol{\lambda}^{\top} \mathbb{E}[\mathbf{g}(\boldsymbol{\pi}(\boldsymbol{\phi};\mathbf{w}),\boldsymbol{\theta})] \quad (9)$$

where λ is the vector of Lagrange multipliers corresponding to the constraints in (8). Vector λ concatenates the multipliers $\overline{\lambda}$ and $\underline{\lambda}$ associated with the lower and upper voltage limits in (3) and (5). A stationary point for the related dual problem

$$D^* := \max_{\boldsymbol{\lambda} \ge \mathbf{0}} \min_{\mathbf{w}: \pi(\boldsymbol{\phi}; \mathbf{w}) \in \mathcal{Q}_{\boldsymbol{\theta}}} L(\mathbf{w}; \boldsymbol{\lambda})$$
(10)

can be obtained iteratively using the primal-dual updates indexed by k (cf. [15]):

$$\mathbf{w}^{k+1} := \begin{bmatrix} \mathbf{w}^k - \mu_w \nabla_{\mathbf{w}} L(\mathbf{w}^k; \boldsymbol{\lambda}^k) \end{bmatrix}_{\mathcal{Q}_{\boldsymbol{\theta}}}$$
(11a)

$$\boldsymbol{\lambda}^{k+1} := \left[\boldsymbol{\lambda}^k + \mu_{\lambda} \nabla_{\boldsymbol{\lambda}} L(\mathbf{w}^{k+1}; \boldsymbol{\lambda}^k)\right]_+$$
(11b)

where (μ_w, μ_λ) are positive step sizes. Here primal variables are updated through projected gradient descent steps on the Lagrangian function. Dual variables are updated through projected gradient ascent steps again on the Lagrangian function. The operator $[x]_+ = \max\{x, 0\}$ is applied entry-wise and ensures $\lambda \ge 0$ at all times.

The operator $[\cdot]_{\mathcal{Q}_{\theta}}$ projects \mathbf{w}^{k+1} such that $\pi(\phi; \mathbf{w}^{k+1}) \in \mathcal{Q}_{\theta}$ for all ϕ . A direct way to confine the DNN output q_n^g within $\pm \sqrt{\overline{s}_n^2 - (p_n^g)^2}$ is to use the hyperbolic tangent (tanh) as the output activation function and then scale the output by $\sqrt{\overline{s}_n^2 - (p_n^g)^2}$. While the apparent power limit \overline{s}_n is known *a priori*, the solar generation p_n is available to the DNN as an input. The required scaling is easily accommodated by minor

architectural modifications to the activation layer of the DNN. Figure 2 illustrates this process for a single-output neuron. Ensuring that $\pi(\phi; \mathbf{w}) \in \mathcal{Q}_{\theta}$ at all times obviates the need for projecting the weight updates henceforth. Note that the gradient $\nabla_{\lambda} L(\mathbf{w}; \lambda)$ in the dual variable update in (11b) can be substituted as $\nabla_{\lambda} L(\mathbf{w}; \lambda) = \mathbf{g}(\pi(\phi; \mathbf{w}), \theta)$.

Following a stochastic approximation approach, the ensemble averages in (9) are first surrogated by sample averages computed over a set of K scenarios $\{\phi^k, \theta^k\}_{k=1}^K$. The average ohmic losses for example can be approximated as

$$\mathbb{E}[\ell(\boldsymbol{\pi}(\boldsymbol{\phi};\mathbf{w}),\boldsymbol{\theta})] \simeq \frac{1}{K} \sum_{k=1}^{K} \ell(\boldsymbol{\pi}(\boldsymbol{\phi}^{k};\mathbf{w}),\boldsymbol{\theta}^{k}).$$

The previous notation captures the fact that the control policy is fed by ϕ to determine **q**, but of course ohmic losses depend on the actual θ . The updates in (11) are then simplified by replacing the sample averages by a single scenario per iteration. In other words, at iteration k, the gradients appearing in (11) are computed for a single scenario k as [15]

$$\mathbf{w}^{k+1} := \mathbf{w}^k - \mu_w \left(\nabla_{\mathbf{w}} \ell^k + \left(\nabla_{\mathbf{w}} \mathbf{g}^k \right)^\top \boldsymbol{\lambda}^k \right)$$
(12a)

$$\boldsymbol{\lambda}^{k+1} := \left[\boldsymbol{\lambda}^k + \mu_{\lambda} \mathbf{g} \left(\boldsymbol{\pi}(\boldsymbol{\phi}^k; \mathbf{w}^{k+1}), \boldsymbol{\theta}^k \right) \right]_+$$
(12b)

where $\nabla_{\mathbf{w}} \ell^k$ is the gradient of ℓ and $\nabla_{\mathbf{w}} \mathbf{g}^k$ is the Jacobian matrix of \mathbf{g} , both with respect to \mathbf{w} and both evaluated at $(\phi^k, \mathbf{w}^k, \theta^k)$. The rest of this section explains how the gradients appearing in (12a) can be computed for the averaged and probabilistic formulations.

A. Averaged Formulation

For the averaged formulation, the stochastic primal-dual updates can be obtained by replacing $g(\pi(\phi; \mathbf{w}), \theta)$ with the constraint functions from (3) to get

$$\mathbf{w}^{k+1} := \mathbf{w}^{k} - \mu_{w} \left(\nabla_{\mathbf{w}} \ell^{k} + \left(\nabla_{\mathbf{w}} \mathbf{v}^{k} \right)^{\top} \left(\overline{\boldsymbol{\lambda}}^{k} - \underline{\boldsymbol{\lambda}}^{k} \right) \right)$$
(13a)

$$\underline{\lambda}^{k+1} := \left[\underline{\lambda}^{k} + \mu_{\lambda} \left(\underline{\mathbf{v}} - \mathbf{v} \left(\pi(\phi^{k}; \mathbf{w}^{k+1}), \boldsymbol{\theta}^{k}\right)\right)\right]_{+} \quad (13b)$$

$$\overline{\boldsymbol{\lambda}}^{k+1} := \left[\overline{\boldsymbol{\lambda}}^k + \mu_{\lambda} \left(\mathbf{v} \left(\boldsymbol{\pi}(\boldsymbol{\phi}^k; \mathbf{w}^{k+1}), \boldsymbol{\theta}^k \right) - \overline{\mathbf{v}} \right) \right]_+.$$
(13c)

To compute the Jacobian matrix $\nabla_{\mathbf{w}} \mathbf{v}$ of voltages with respect to DNN weights, we resort to the power flow equations. Let vector $\tilde{\mathbf{u}} \in \mathbb{C}^{N+1}$ collect the complex voltage phasors at all buses and define $\bar{\mathbf{u}} := [\operatorname{Re}(\tilde{\mathbf{u}})^\top \operatorname{Im}(\tilde{\mathbf{u}})^\top]^\top \in \mathbb{R}^{2N+2}$. Vector $\mathbf{u} \in \mathbb{R}^{2N}$ is obtained by dropping the first and (N+1)th entries of $\bar{\mathbf{u}}$. These two entries correspond to the substation voltage \tilde{u}_0 , which is assumed constant. The sought Jacobian matrix can be computed via the chain rule as

$$\nabla_{\mathbf{w}}\mathbf{v} = \nabla_{\mathbf{u}}\mathbf{v} \cdot \nabla_{\mathbf{q}}\mathbf{u} \cdot \nabla_{\mathbf{w}}\mathbf{q}.$$
 (14)

We next elaborate on the three Jacobian matrices needed in (14). The $N \times 2N$ matrix $\nabla_{\mathbf{u}} \mathbf{v}$ can be readily computed by definition of voltage magnitudes. Its (n, m)-th entry is

$$[\nabla_{\mathbf{u}}\mathbf{v}]_{n,m} = \begin{cases} \frac{u_n}{v_n} &, \ m = n\\ \frac{u_{n+N}}{v_n} &, \ m = n+N\\ 0 &, \ \text{otherwise.} \end{cases}$$

We proceed with finding $\nabla_{\mathbf{q}} \mathbf{u}$. If $\mathbf{p} + j\mathbf{q}$ is the vector of complex power injections at all buses excluding the substation, define $\mathbf{s} := [\mathbf{p}^\top \mathbf{q}^\top]^\top$. Per the power flow equations, every entry *i* of \mathbf{s} can be expressed as a quadratic function of $\bar{\mathbf{u}}$, that is $s_i = \bar{\mathbf{u}}^\top \mathbf{M}_i \bar{\mathbf{u}}$ for a symmetric real-valued matrix \mathbf{M}_i derived from the bus admittance matrix of the feeder; see e.g., [22] for the detailed expressions for \mathbf{M}_i 's. Therefore, the *i*-th row of the Jacobian matrix $\nabla_{\bar{\mathbf{u}}} \mathbf{s}$ is given by $2\bar{\mathbf{u}}^\top \mathbf{M}_i$. By dropping the first and (N + 1)-th column of $\nabla_{\bar{\mathbf{u}}} \mathbf{s}$, we obtain $\nabla_{\mathbf{u}} \mathbf{s}$. Under mild technical conditions, the inverse function theorem predicates that

$$\nabla_{\mathbf{s}} \mathbf{u} = (\nabla_{\mathbf{u}} \mathbf{s})^{-1}. \tag{15}$$

Matrix $\nabla_{\mathbf{q}}\mathbf{u}$ can be clearly obtained by keeping only the last N rows of $\nabla_{\mathbf{s}}\mathbf{u}$. The third matrix $\nabla_{\mathbf{w}}\mathbf{q}$ can be readily computed using gradient back-propagation across the DNN.

The gradient vector of ohmic losses with respect to the DNN weights can be computed similarly as

$$(\nabla_{\mathbf{w}}\ell)^{\top} = (\nabla_{\mathbf{u}}\ell)^{\top} \cdot \nabla_{\mathbf{q}}\mathbf{u} \cdot \nabla_{\mathbf{w}}\mathbf{q}.$$
 (16)

The gradient $\nabla_{\mathbf{u}}\ell$ can be easily computed by recognizing

$$\ell = \sum_{n=0}^{N} p_n = \bar{\mathbf{u}}^\top \left(\sum_{n=0}^{N} \mathbf{M}_i\right) \bar{\mathbf{u}}$$

where matrix \mathbf{M}_0 describes the power injection at the substation as $p_0 = \bar{\mathbf{u}}^\top \mathbf{M}_0 \bar{\mathbf{u}}$ similar to the remaining injections. It is then obvious that $\nabla_{\bar{\mathbf{u}}} \ell = 2 \sum_{n=0}^{N} \mathbf{M}_i \bar{\mathbf{u}}$. The gradient $\nabla_{\mathbf{u}} \ell$ of (16) is found by dropping the first and (N + 1)-th entries of vector $\nabla_{\bar{\mathbf{u}}} \ell$.

B. Probabilistic Formulation

For the probabilistic formulation of (5), the update steps in (12) cannot be applied directly. This is because constraints (5b)–(5c) involve the indicator function that is not differentiable, and so the Jacobian matrix $\nabla_{\mathbf{w}}\mathbf{g}$ does not exist. Moreover, these constraints are non-convex, thus prohibiting stochastic (sub)gradient updates. To circumvent these complications, we instead turn to the convex CVaR approximations to (5) using stochastic subgradient primal-dual updates.

We first briefly review the CVaR approximation of chance constraints from [18], and then compute the related subgradients. For some $\alpha \in (0,1)$, consider the chance constraint $\Pr[f_{\theta}(x) \geq 0] = \mathbb{E}_{\theta}[\mathbb{1}(f_{\theta}(x))] \leq \alpha$, where function f depends on the optimization variable x and a random variable θ . Note that $\mathbb{1}(f_{\theta}(x)) \leq [1 + f_{\theta}(x)/t]_{+}$ for all $f_{\theta}(x)$ and t > 0. This is easy to verify by checking the two cases in the definition of the indicator function in (6). Then, if there exists a t > 0 satisfying $\mathbb{E}_{\theta} \left[[1 + f_{\theta}(x)/t]_{+} \right] \leq \alpha$, the original chance constraint $\mathbb{E}_{\theta}[\mathbb{1}(f_{\theta}(x))] \leq \alpha$ holds too. Since t > 0, the restriction of the chance constraint can be alternatively expressed as $\mathbb{E}_{\theta}[[t + f_{\theta}(x)]_{+}] < \alpha t$. In fact, the requirement t > 0 can be dropped because for all negative t, the constraint $\mathbb{E}_{\theta}[t+f_{\theta}(x)]_{+} \leq \alpha t$ becomes infeasible. And for t=0, the restricted constraint yields $\mathbb{E}_{\theta}\left[\left[f_{\theta}(x)\right]_{+}\right] \leq 0$ or equivalently $f_{\theta}(x) < 0$ for all θ , so that the original chance constraint holds trivially. Therefore, imposing the convex constraint

 $\mathbb{E}_{\theta}[[t+f_{\theta}(x)]_+] \leq \alpha t$ for some t constitutes a restriction of the original chance constraint $\Pr[f_{\theta}(x) \geq 0] \leq \alpha$.

By identifying $f_{\theta}(x)$ with $\underline{v}_n - v_n(\mathbf{q}, \theta)$ and introducing an auxiliary variable \underline{t}_n per bus *n*, we can now restrict the voltage chance constraint in (5b) by imposing constraint

$$\mathbb{E}\left[\left[\underline{t}_n + \underline{v}_n - v_n(\mathbf{q}, \boldsymbol{\theta})\right]_+\right] \le \alpha \underline{t}_n, \ \forall n \in \mathcal{N}.$$

The chance constraints of (5c) on upper voltage limits can be treated similarly using variables \bar{t}_n 's. Collecting auxiliary variables $\{\underline{t}_n, \bar{t}_n\}_{n=1}^N$ in vectors \underline{t} and \overline{t} accordingly, we can now formulate the convex restriction of (5) as

$$\min_{\mathbf{q}\in\mathcal{Q}_{\boldsymbol{\theta},\mathbf{t},\bar{\mathbf{t}}}} \mathbb{E}[\ell(\mathbf{q},\boldsymbol{\theta})]$$
(17a)

s.to
$$\mathbb{E}\left[\left[\underline{\mathbf{t}} + \underline{\mathbf{v}} - \mathbf{v}(\mathbf{q}, \boldsymbol{\theta})\right]_{+} - \alpha \underline{\mathbf{t}}\right] \leq \mathbf{0}$$
 (17b)

$$\mathbb{E}\left[\left[\mathbf{t} + \mathbf{v}(\mathbf{q}, \boldsymbol{\theta}) - \overline{\mathbf{v}}\right]_{+} - \alpha \mathbf{t}\right] \le \mathbf{0}.$$
(17c)

For brevity, let the expressions inside the expectation operator of (17b)–(17c) be represented by $\underline{\mathbf{g}}(\underline{\mathbf{t}}, \mathbf{v}(\mathbf{q}, \boldsymbol{\theta}))$ and $\overline{\mathbf{g}}(\overline{\mathbf{t}}, \mathbf{v}(\mathbf{q}, \boldsymbol{\theta}))$. Similar to (8), problem (17) can be tackled using stochastic primal-dual updates upon replacing the ensemble with sample averages over K scenarios. Different from the averaged formulation however, the constraint functions $\underline{\mathbf{g}}(\underline{\mathbf{t}}, \mathbf{v}(\mathbf{q}, \boldsymbol{\theta}))$ and $\overline{\mathbf{g}}(\overline{\mathbf{t}}, \mathbf{v}(\mathbf{q}, \boldsymbol{\theta}))$ are non-differentiable with respect to $(\mathbf{v}, \underline{\mathbf{t}}, \overline{\mathbf{t}})$. We use their subgradients instead.

$$\mathbf{w}^{k+1} := \mathbf{w}^{k} - \mu_{w} \left(\nabla_{\mathbf{w}} \ell^{k} + \left(\partial_{\mathbf{w}} \underline{\mathbf{g}}^{k} \right)^{\top} \underline{\boldsymbol{\lambda}}^{k} + \left(\partial_{\mathbf{w}} \overline{\mathbf{g}}^{k} \right)^{\top} \overline{\boldsymbol{\lambda}}^{k} \right)$$
(18a)

$$\underline{\mathbf{t}}^{k+1} := \underline{\mathbf{t}}^k - \mu_t \left(\partial_{\underline{\mathbf{t}}} \underline{\mathbf{g}}^k \right)^\top \underline{\boldsymbol{\lambda}}^k \tag{18b}$$

$$\overline{\mathbf{t}}^{k+1} := \overline{\mathbf{t}}^k - \mu_t \left(\partial_{\overline{\mathbf{t}}} \overline{\mathbf{g}}^k \right)^\top \overline{\boldsymbol{\lambda}}^k \tag{18c}$$

$$\underline{\boldsymbol{\lambda}}^{k+1} := \left[\underline{\boldsymbol{\lambda}}^{k} + \mu_{\lambda} \underline{\mathbf{g}} \left(\underline{\mathbf{t}}^{k+1}, \mathbf{v} \left(\mathbf{q}^{k+1}, \boldsymbol{\theta}^{k} \right) \right) \right]_{+}$$
(18d)

$$\overline{\boldsymbol{\lambda}}^{k+1} := \left[\overline{\boldsymbol{\lambda}}^k + \mu_{\lambda} \overline{\mathbf{g}} \left(\overline{\mathbf{t}}^{k+1}, \mathbf{v} \left(\mathbf{q}^{k+1}, \boldsymbol{\theta}^k \right) \right) \right]_+.$$
(18e)

To compute the needed subgradients, recall that a subgradient of $f(x) = [x]_+$ can be found as

$$\partial f(x) = \begin{cases} 0 & , \ x < 0 \\ 1 & , \ x \ge 0 \end{cases} = \mathbb{1}(x).$$

The subgradients involved in (18b)–(18c) can be computed using the chain rule as

$$\partial_{\underline{\mathbf{t}}} \underline{\mathbf{g}} = \operatorname{dg} \left(\mathbb{1} \left(\underline{\mathbf{t}} + \underline{\mathbf{v}} - \mathbf{v} \right) \right) - \alpha \mathbf{I}_N$$
$$\partial_{\overline{\mathbf{t}}} \overline{\mathbf{g}} = \operatorname{dg} \left(\mathbb{1} \left(\overline{\mathbf{t}} + \mathbf{v} - \overline{\mathbf{v}} \right) \right) - \alpha \mathbf{I}_N$$

where the indicator functions here are applied entrywise and evaluate to vectors. In turn, the subgradients appearing in (18a) can be found as

$$\partial_{\mathbf{w}} \mathbf{g} = \partial_{\mathbf{v}} \mathbf{g} \cdot \nabla_{\mathbf{w}} \mathbf{v} \quad \text{and} \quad \partial_{\mathbf{w}} \overline{\mathbf{g}} = \partial_{\mathbf{v}} \overline{\mathbf{g}} \cdot \nabla_{\mathbf{w}} \mathbf{v}.$$

The Jacobian $\nabla_{\mathbf{w}} \mathbf{v}$ has already been computed in (14), while the subgradients with respect to voltage magnitudes are

$$\partial_{\mathbf{v}} \underline{\mathbf{g}} = -\operatorname{dg}\left(\mathbbm{1}\left(\underline{\mathbf{t}} + \underline{\mathbf{v}} - \mathbf{v}\right)\right) \\ \partial_{\mathbf{v}} \overline{\mathbf{g}} = \operatorname{dg}\left(\mathbbm{1}\left(\overline{\mathbf{t}} + \mathbf{v} - \overline{\mathbf{v}}\right)\right).$$

IV. GRADIENT-FREE IMPLEMENTATION

To compute the (sub)gradients appearing in the primal-dual updates of Section III, an accurate model of the distribution feeder (including its connectivity and line impedances) is required. Such requirement may not be realistic under two scenarios: *i*) If the utility does not have a detailed feeder model; and *ii*) When inverters are controlled by a third-party entity such as an aggregator with which the utility is reluctant to share feeder models due to security and privacy limitations. This section puts forth a gradient-free implementation of the DNN updates relying on a *digital twin* that is a black-box or query model of the feeder. Once fed with grid conditions θ and inverter setpoints **q**, the digital twin provides the vector of nodal voltages **v** and ohmic losses ℓ as its outputs. Through this model, an aggregator can map injections to voltages and losses yet without knowing the precise feeder model.

Lacking the feeder physical model precludes the sensitivity calculations derived using the power flow equations. In other words, one cannot compute the partial derivatives appearing in the left-hand side of (14) and (16). Nonetheless, we can aim directly for the Jacobian matrix $\nabla_{\mathbf{q}} \mathbf{v}$ and the gradient vector $\nabla_{\mathbf{q}} \ell$, and approximate them through finite differences. In detail, we resort to zeroth-order approximants (see [23]) of the needed sensitivities by querying the digital twin twice to obtain two function evaluations as:

$$\hat{\nabla}_{\mathbf{q}}\ell = \frac{\ell(\mathbf{q} + \epsilon\check{\mathbf{q}}, \boldsymbol{\theta}) - \ell(\mathbf{q} - \epsilon\check{\mathbf{q}}, \boldsymbol{\theta})}{2\epsilon}\check{\mathbf{q}}^{\top}$$
(19a)

$$\hat{\nabla}_{\mathbf{q}}\mathbf{v} = \frac{\mathbf{v}(\mathbf{q} + \epsilon\check{\mathbf{q}}, \boldsymbol{\theta}) - \mathbf{v}(\mathbf{q} - \epsilon\check{\mathbf{q}}, , \boldsymbol{\theta})}{2\epsilon}\check{\mathbf{q}}^{\top}$$
(19b)

where ϵ is the scale of perturbation, and $\check{\mathbf{q}}$ is a perturbation vector Gaussian distributed with zero-mean and standard deviation $\sigma_{\check{\mathbf{q}}}$. The quantities ϵ and $\sigma_{\check{\mathbf{q}}}$ are treated as hyperparameters and are set during the training process.. The approximations in (19) are carried out in three steps. First, the DNN is presented with the input ϕ and its output \mathbf{q} is recorded. Second, the digital twin is presented with (\mathbf{q}, θ) and computes $\ell(\mathbf{q}, \theta)$ and $\mathbf{v}(\mathbf{q}, \theta)$. Third, the digital twin is presented with $(\mathbf{q} + \epsilon \check{\mathbf{q}}, \theta)$ and computes $\mathbf{v}(\mathbf{q} + \epsilon \check{\mathbf{q}}, \theta)$. Fig 3 compares the steps for obtaining the quantities $\nabla_{\mathbf{w}} \ell$ and $\nabla_{\mathbf{w}} \mathbf{v}$ for the gradient-based and gradient-free approaches.

With the finite-difference approximants of (19) in place, the *gradient-free* primal-dual updates are straightforward to execute by approximating

$$(\hat{\nabla}_{\mathbf{w}}\ell)^{\top} = (\hat{\nabla}_{\mathbf{q}}\ell)^{\top}\nabla_{\mathbf{w}}\mathbf{q}$$
$$\hat{\nabla}_{\mathbf{w}}\mathbf{v} = \hat{\nabla}_{\mathbf{q}}\mathbf{v}\cdot\nabla_{\mathbf{w}}\mathbf{q}$$

where $\nabla_{\mathbf{w}} \mathbf{q}$ is again calculated gradient back-propagation.

V. NUMERICAL TESTS

The performance of the proposed DNN-based control strategy was evaluated using a single-phase version of the IEEE 37-bus feeder. Real-world one-minute active load and solar generation data were extracted for April 2, 2011 from the Smart* project [24], [25]. For active loads, homes with IDs 20-369 were used. Averaged load demands were calculated by considering 10 homes at a time, and were serially allotted to



Fig. 3. Steps for calculating $\nabla_{\mathbf{w}} \ell$ and $\nabla_{\mathbf{w}} \mathbf{v}$ for the *gradient-based* (top) and *gradient-free* (bottom) approaches. The *gradient-based* approach requires a solution to the power flow equations in conjunction with the inverse function theorem (Section III); the *gradient-free* obtains the desired quantities by probing the digital twin and applying zero-order approximations.



Fig. 4. The IEEE 37-bus feeder used for the numerical tests. Node numbering follows the format node number {panel ID}. The inverters at nodes $\{12, 20, 22, 24, 25\}$ provide reactive power control, whereas the rest operate at unit power factor.

buses 2-36 of Fig. 4. The values of active loads were scaled so their maximum active load per node matched its benchmark value. Reactive loads were then added to each of these homes by sampling lagging power factors uniformly within [0.9, 1.0]and for each time interval. For solar generation, panel IDs were matched to the buses as shown in Fig. 4. Solar generation values were scaled so the maximum generation per panel was 2 times the benchmark value. Out of all the nodes with inverterinterfaced solar generation, those at nodes $\{12, 20, 22, 24, 25\}$ were also providing reactive power support. The extracted data points were considered as available forecasts for 4-hour control periods. Appropriate training and testing scenarios were created. Zero-mean white Gaussian noise was added to the 240 one-minute data points from the forecast to create a total of 1200 samples. The standard deviation of the Gaussian noise was set to 0.1 times the mean load forecast. Out of the 1200 samples, 960 were used as the training set and the remaining 240 formed the testing set. The training samples were additionally randomly shuffled to promote better generalization for the DNNs.

All tests were conducted on a 2.4 GHz 8-Core Intel Core i9 processor laptop computer with 64 GB RAM. Simulation scripts were written in Python and TensorFlow libraries to implement and train the DNNs. For the tests presented, fourlayered fully connected DNNs were employed. The grid conditions vector $\boldsymbol{\theta} := [\mathbf{p}^g, \mathbf{p}^c, \mathbf{q}^c]^{\top}$ consisting of measurements from the M < N buses equipped with smart meters were fed as inputs to the DNNs. Therefore, the input layers were chosen to have 3M neurons. The two subsequent hidden layers were fixed to having 3N and 2N neurons, respectively. Finally, the output layers had 5 neurons corresponding to the 5 inverters. All but the final layers of the DNNs employed the ReLU (rectified linear unit) activation with the final layers using a scaled *tanh* activation to ensure the inverter limits $q^g \in Q^t$. The weights for the DNN layers were initiated from a Gaussian distribution with zero mean and a unit standard deviation. The biases for the DNN layers, the dual variables, and the auxiliary variables were all initialized at zero. Additional modelling and training details are presented along with the discussions of the results as follows.

A. Averaged Formulation

For the average formulation, the DNN was fed with the complete vector of grid conditions θ obtained from measurements collected at all buses. DNN weights were updated using Adam with a learning rate of 0.001. Dual variables were updated using SGD with a learning rate of 10 that decayed with the square-root of the iteration index [26]. The model was then trained for 15 epochs over the training scenarios.

To demonstrate the efficacy of the proposed approach, the results are compared against a no compensation scenario, which represents the grid conditions in the absence of reactive power compensation by inverters. The DNN-based approach is also benchmarked against an optimal approach that solves the deterministic problem in (2) per minute. As discussed previously, such optimal approach might not be realistic to implement in real time due to the high computational burden. Fig. 5 compares the average losses and bus voltages under the three scenarios over the training set and during the high solar period of 12-4 pm. Without any reactive power compensation, buses {18, 19, 20, 21, 22, 33, 34} experience overvoltages. The proposed DNN-based approach behaves as expected by lowering the average voltages at these buses down to the acceptable range. The optimal approach also achieves the same objective but by bringing all instantaneous voltages to the desired range whenever feasible. Note that both the DNN-based approach and the optimal approach incur higher losses when compared to the no compensation scenario. This is a result of increase in the magnitude of line currents on account of reactive power withdrawals. Since the optimal approach focuses on instantaneous voltage values rather than



Fig. 5. *Top:* Time-averaged losses during the 12–4 pm testing interval attained by the instantaneously optimal control strategy of (2); the proposed DNN-based inverter control; and no reactive power compensation by inverters. *Bottom:* Box plots showing the first and third quantiles of the voltage deviations experienced across buses under the three control strategies. The proposed strategy achieves lower average losses over optimal inverter control as voltages are not constrained within $\pm 3\%$ at all times.



Fig. 6. Results for averaged formulation over testing data during the interval 12–4 pm: Average losses under the instantaneous optimal strategy, the proposed DNN-based approach; and no reactive power compensation are depicted on the top panel. Voltage deviations across buses under the three strategies are shown at the bottom panel.

their averages, it incurs higher losses when compared to the DNN-based approach. The trained DNN was then evaluated over unseen scenarios of the testing set. As can be seen in Fig. 6, the proposed approach performed remarkably well in maintaining voltages within limits and lowering average losses over the testing set.

Finally, the sample probabilities for voltage violations were calculated for the resulting voltages from the no compensation case and the proposed DNN-based approaches. For the buses with non-zero values for these probabilities, radar plots were drawn as shown in Fig. 7. The radial-axis represents the values



Fig. 7. Voltage profiles during the 12–4 pm interval for averaged formulation. Results under no reactive power compensation and under the proposed DNN-based policies have been compared. Angular markings correspond to bus numbers, whereas radial markings are sampled probabilities of voltage violations.



Fig. 8. Results for probabilistic formulations for the 12–4 pm window. Voltage profiles for different values of $\alpha = \{0.7, 0.5, 0.3\}$ are depicted. Angular markings correspond to bus numbers whereas radial markings are sampled probabilities of voltage limits violation.

for sample probabilities for voltage limit violations, whereas the angular markings correspond to the bus numbers. One can see that without any reactive power support, the grid faces a high probability of voltage violations over both training and testing. While the average formulation successfully regulates the average voltages, its effect on reducing the total number of occurrences of voltage violations is somewhat modest with buses $\{19, 20, 21, 22, 23\}$ violating the voltage limits for more than half of the scenarios.

B. Probabilistic Formulation

Employing the DNN architecture from the previous subsection and the full input vector θ , updates in (18) were applied to train the DNN for the probabilistic formulation. The DNN weights were updated using Adam with a learning rate of 0.001. For the auxiliary variables $\{\underline{t}, \overline{t}\}$, Adam optimizer with a learning rate of 0.001 was deployed, and the dual variables were updated using SGD with a learning rate of 1 that decayed with the square-root of the iteration index. The model required a higher number of 20 epochs to converge during training because of the additional primal variables t. The experiments were conducted for the same time period of 12–4 pm. The experiments were repeated for three different values of $\alpha = \{0.7, 0.5, 0.3\}$ and the radar plots for the resulting sample probabilities of voltage violations are shown in Fig. 8.



Fig. 9. Mean voltage deviations across the buses under gradient-based and gradient-free approaches for the averaged formulation and during the testing 12–4 pm interval.

As desired, when compared to the averaged formulation, the occurrences of voltage violations under the probabilistic formulation were found to be drastically less for lower values of α . Since the calculated sample probabilities came out to be less than the set α , the results in Fig. 8 confirm the conservative nature (restriction) of the convex approximations in (17).

C. Gradient-free implementation

To quantify the accuracy of the gradient approximations in (19), the DNNs for the averaged and probabilistic formulations were trained under the gradient-free fashion. The scale of perturbation ϵ was set to 0.1, and the perturbations $\check{\mathbf{q}}$ were sampled from a zero-mean Gaussian distribution with a unit standard deviation. The gradient-free approaches were compared to their gradient-based counterparts over the same time periods and the results shown in Figs. 9-10.

Fig. 9 shows the mean voltage deviations across all the buses and time under the two approaches. The gradient-free approach incurs slightly higher voltage violations, but otherwise matches the performance of the gradient-based approach surprisingly well without any explicit knowledge of the underlying relationships. Similar results were confirmed for the probabilistic formulation in Fig. 10, where both approaches yield similar sample probabilities for voltage violations over training and testing, with $\alpha = 0.5$.

D. Partial inputs

To study the effect of partial DNN inputs on the control performance of the learned DNN policy, we varied the number of nodes whose data are telemetered in real time for the probabilistic formulation. First, real-time meters were assigned to all nodes with solar generation. Then, different input scenarios were simulated by expanding the subset of inverter-equipped nodes with real-time metering moving from nodes 2–11; nodes 2–16; nodes 2–21; and a full input vector consisting of all nodes. The mean-sampled probabilities of voltage violations across time and buses were recorded. The value of α was set to 0.5 for all scenarios. Figures 11 and 12 show the



Fig. 10. Comparison of gradient-based and gradient-free approaches for the probabilistic formulation with $\alpha = 0.5$ over the testing 12–4 pm window. Voltage profiles for both approaches are depicted using polar plots.



Fig. 11. Mean sampled probabilities of voltage violations for the probabilistic formulation with $\alpha = 0.5$ during training over 12–4 pm. The dimension of the DNN input increases as the number of nodes monitored in real time increases.

results recorded respectively during training and testing. The results confirm the intuition that the performance of the control policies improves as more real-time information is provided to the DNN. As the nodes with real-time monitoring increase, the sampled probabilities of voltage violations decrease.

VI. CONCLUSIONS

This work has presented a DNN-based approach for stochastic optimal inverter control. To capture uncertainty, the grid conditions have been modeled as random variables and the associated inverter setpoints by a stochastic policy learned through a DNN. The DNN is periodically trained offline to minimize the average ohmic losses and maintain either the average voltages within limits or the probability of voltage deviation occurrences low. Training is accomplished by adopting existing stochastic primal-dual updates to the AC OPF setup. The proposed scheme not only expedites the computation of near-optimal inverter setpoints, but also resolves two practical difficulties: d1) How to solve an OPF using only a power flow solver or a digital twin of the feeder? and d2) How to deal with an OPF if grid conditions are only partially known?



Fig. 12. Mean sampled probabilities of voltage violations for the probabilistic formulation with $\alpha = 0.5$ during testing over 12–4 pm. The dimension of the DNN input increases as the number of nodes monitored in real time increases.

Numerical tests on the benchmark IEEE 37-bus feeder showcase the salient features of the novel methodology. The adopted stochastic primal-dual updates train a DNN-based policy using a modest number of training samples. The DNN is numerically shown to produce an inverter dispatch that minimizes the OPF cost while satisfying the operating constraints. Although the averaged formulation succeeds in maintaining the voltage at each node within limits on the average, violations do occur frequently. To that end, the chance-constrained formulation may be more relevant. The latter comes at minimal extra complexity over the averaged formulation. Being a convex restriction the numerical tests corroborate that it is a conservative yet safe scheme. Our experiments have also shown how the DNN-based policy can be driven with incomplete grid conditions, and demonstrated the improvement in feasibility when more information is presented to the DNN. A final interesting outcome is that when the DNNs are trained using the gradient-free updates, the degradation in performance is minimal although the learner has less information about the feeder at its disposal.

Some open questions are to: *i*) Extend the implementation to a multiphase grid model with detailed ZIP loads, regulators, and capacitor banks; *ii*) Consider additional network constraints and/or alternative objectives; *iii*) Experiment with the frequency of re-training the DNN, the size of the training dataset, and the way it has been generated; *iv*) Utilize the DNN output only to warm-start an actual OPF solver; and *v*) Optimally select the grid proxies to improve inverter control performance under a communication budget.

REFERENCES

- M. Farivar, R. Neal, C. Clarke, and S. Low, "Optimal inverter VAR control in distribution systems with high PV penetration," in *Proc. IEEE Power & Energy Society General Meeting*, San Diego, CA, Jul. 2012, pp. 1–7.
- [2] G. Wang, V. Kekatos, A. J. Conejo, and G. B. Giannakis, "Ergodic energy management leveraging resource variability in distribution grids," *IEEE Trans. Power Syst.*, vol. 31, no. 6, pp. 4765–4775, Nov. 2016.

- [3] S. Karagiannopoulos, P. Aristidou, and G. Hug, "Data-driven local control design for active distribution grids using off-line optimal power flow and machine learning techniques," *IEEE Trans. Smart Grid*, vol. 10, no. 6, pp. 6461–6471, Nov. 2019.
- [4] 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.
- [5] 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.
- [6] N. Guha, Z. Wang, M. Wytock, and A. Majumdar, "Machine learning for AC optimal power flow," in *Climate Change Workshop at ICML*, Long Beach, CA, Jun. 2019, pp. 1–4.
- [7] Y. Yang, Z. Yang, J. Yu, B. Zhang, Y. Zhang, and H. Yu, "Fast calculation of probabilistic power flow: A model-based deep learning approach," *IEEE Trans. Smart Grid*, vol. 11, no. 3, pp. 2235–2244, May 2020.
- [8] D. Owerko, F. Gama, and A. Ribeiro, "Optimal power flow using graph neural networks," in *Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, Barcelona, Spain, May 2020, pp. 5930–5934.
- [9] M. K. Singh, S. Gupta, V. Kekatos, G. Cavraro, and A. Bernstein, "Learning to optimize power distribution grids using sensitivityinformed deep neural networks," in *Proc. IEEE Intl. Conf. on Smart Grid Commun.*, Tempe, AZ, Nov. 2020, pp. 1–6.
- [10] M. K. Singh, V. Kekatos, and G. B. Giannakis, "Learning to solve the AC-OPF using sensitivity-informed deep neural networks," *IEEE Trans. Power Syst.*, 2021, submitted. [Online]. Available: https://arxiv.org/abs/2103.14779
- [11] M. Jalali, V. Kekatos, N. Gatsis, and D. Deka, "Designing reactive power control rules for smart inverters using support vector machines," *IEEE Trans. Smart Grid*, vol. 11, no. 2, pp. 1759–1770, Mar. 2020.
- [12] Q. Yang, G. Wang, A. Sadeghi, G. Wang, G. B. Giannakis, and J. Sun, "Real-time voltage control using deep reinforcement learning," *IEEE Trans. Smart Grid*, vol. 11, no. 3, pp. 2313–2323, May 2020.
- [13] Q. Zhang, K. Dehghanpour, Z. Wang, F. Qiu, and D. Zhao, "Multi-agent safe policy learning for power management of networked microgrids," *IEEE Trans. Smart Grid*, vol. 12, no. 2, pp. 1048–1062, Mar. 2021.
- [14] 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*, vol. 11, no. 4, pp. 3008–3018, Jul. 2020.
- [15] M. Eisen, C. Zhang, L. F. O. Chamon, D. D. Lee, and A. Ribeiro,

"Learning optimal resource allocations in wireless systems," *IEEE Trans. Signal Processing*, vol. 67, no. 10, pp. 2775–2790, May 2019.

- [16] S. Gupta, V. Kekatos, and M. Jin, "Machine learning for communicationcognizant smart inverter control," in *Proc. IEEE Intl. Conf. on Smart Grid Commun.*, Tempe, AZ, Nov. 2020, pp. 1–6.
- [17] S. Taheri, M. Jalali, V. Kekatos, and L. Tong, "Fast probabilistic hosting capacity analysis for active distribution systems," *IEEE Trans. Smart Grid*, 2020, (submitted). [Online]. Available: https: //arxiv.org/abs/2002.01980
- [18] A. Nemirovski and A. Shapiro, "Convex approximations of chance constrained programs," *SIAM Journal on Optimization*, vol. 17, no. 4, pp. 969–996, 2007.
- [19] R. A. Jabr, "Linear decision rules for control of reactive power by distributed photovoltaic generators," *IEEE Trans. Power Syst.*, vol. 33, no. 2, pp. 2165–2174, Mar. 2019.
- [20] W. Lin and E. Bitar, "Decentralized stochastic control of distributed energy resources," *IEEE Trans. Power Syst.*, vol. 33, no. 1, pp. 888– 900, Jan. 2018.
- [21] K. Baker, A. Bernstein, E. Dall'Anese, and C. Zhao, "Network-cognizant voltage droop control for distribution grids," *IEEE Trans. Power Syst.*, vol. 33, no. 2, pp. 2098–2108, Mar. 2018.
- [22] V. Kekatos, G. Wang, H. Zhu, and G. B. Giannakis, "PSSE redux: Convex relaxation, decentralized, robust, and dynamic approaches," in Advances in Electric Power and Energy, M. El-Hawary, Ed. Wiley IEEE Press, 2018. [Online]. Available: http://www.faculty.ece.vt.edu/ kekatos/papers/PSSE-Redux.pdf
- [23] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 527–566, 2017.
- [24] 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 *Workshop on Data Mining Applications in Sustainability*, Beijing, China, Aug. 2012, pp. 1–6.
- [25] D. Chen, S. Iyengar, D. Irwin, and P. Shenoy, "Sunspot: Exposing the location of anonymous solar-powered homes," in ACM International Conference on Systems for Energy-Efficient Built Environments, Palo Alto, CA, Nov 2016, p. 85–94.
 [26] L. M. Lopez-Ramos, V. Kekatos, A. G. Marques, and G. B. Giannakis,
- [26] L. M. Lopez-Ramos, V. Kekatos, A. G. Marques, and G. B. Giannakis, "Two-timescale stochastic dispatch of smart distribution grids," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 4282–4292, Sep. 2018.