Deep Learning for Reactive Power Control of Smart Inverters under Communication Constraints

Sarthak Gupta, Vassilis Kekatos, and Ming Jin
Bradley Dept. of Electrical & Computer Engineering
Virginia Tech, Blacksburg, VA 24061, USA
Emails: {gsarthak,kekatos,jinming}@vt.edu

Abstract—Aiming for the median solution between cyber-intensive optimal power flow (OPF) solutions and subpar local control, this work advocates deciding inverter injection setpoints using deep neural networks (DNNs). Instead of fitting OPF solutions in a black-box manner, inverter DNNs are naturally integrated with the feeder model and trained to minimize a grid-wide objective subject to inverter and network constraints enforced on the average over uncertain grid conditions. Learning occurs in a quasi-stationary fashion and is posed as a stochastic OPF, handled via stochastic primal-dual updates acting on grid data scenarios. Although trained as a whole, the proposed DNN is operated in a master-slave architecture. Its master part is run at the utility to output a condensed control signal broadcast to all inverters. Its slave parts are implemented by inverters and are driven by the utility signal along with local inverter readings. This novel DNN structure uniquely addresses the small-big data conundrum where utilities collect detailed smart meter readings yet on an hourly basis, while in real time inverters should be driven by local inputs and minimal utility coordination to save on communication. Numerical tests corroborate the efficacy of this physics-aware DNN-based inverter solution over an optimal control policy.

Index Terms—Neural networks; voltage regulation; power loss minimization; optimal power flow.

I. INTRODUCTION

Distribution grids are currently challenged by voltage fluctuations due to the proliferation of distributed energy resources (DERs). The voltages experienced at buses of a feeder depend heavily on the power injected or withdrawn, while the power generated by a PV under intermittent cloud coverage may vary by 80% within one-minute intervals [1]. The inverters interfacing DERs have been suggested as a promising fast-responding mechanism and are now allowed to provide reactive power support per the amended IEEE 1547 standard. If properly orchestrated, inverters can regulate nodal voltages and/or reduce ohmic line losses. Nonetheless, coordinating hundreds of inverters in real-time is a formidable task.

The literature on inverter control can be broadly classified into optimization- and learning-based approaches. The former class includes approaches where inverter control is posed as an optimal power flow (OPF) problem. Under a centralized OPF setup [2], [3], the utility reads the values of solar generation and loads, solves an OPF, and communicates the optimal setpoints to inverters. To avoid any cyber overhead, inverter setpoints can be decided using simple Volt/VAR or Watt/VAR control rules driven by local readings [1]. Nonetheless, the equilibria of such rules do not coincide with the sought OPF solutions and can be subpar [4], [5].

Learning-based approaches shift the computational effort offline, and perform numerically less intensive tasks during real-time operation. Learning-based approaches can be further clustered into the *OPF-then-learn* and the *OPF-and-learn* philosophies. According to the former, one first solves a large number of OPF instances parameterized by their inputs (solar/load conditions). The pairs of OPF inputs or instances and OPF minimizers are subsequently used for the ML model to learn the OPF mapping in a supervised manner. In real time, the ML model approximates OPF decisions on the fly as soon as it is presented with a new OPF instance. Under this paradigm, references [6] and [7] use kernel–based regression to learn inverter control rules. DNNs have alternatively been employed to learn OPF solutions under a linearized [8]; or an exact AC grid model [9], [10], [11], [12].

Rather than fitting OPF minimizers, the *OPF-and-learn* paradigm trains an ML model directly through an OPF in a single step. Therefore, it does not require solving multiple OPFs to generate a labeled training set. Under the *OPF-and-learn* paradigm, reference [13] adopts kernel-based learning to design inverter control rules, adjusted to grid conditions in a quasi-stationary fashion. Although rules can be learned using a convex program, the kernel functions have to be specified beforehand. In [14], inverter control rules are optimized along with capacitor status decisions to minimize voltage deviations using a two-timescale reinforcement learning (RL) approach. Nonetheless, no feeder-level constraints are involved. Enforcing network constraints is challenging for learning-based OPF methods. One could heuristically project the ML prediction for the OPF solution [9], [13]. Other approaches to coping with constraints include penalizing constraint deviations [7], [8], [13], [14]; or enforcing constraints in a discounted sense [15]. Reference [15] models inverter policies as DNNs. It successively linearizes feeder constraints and updates policies continuously through communication exchanges between interconnected microgrids. A similar *safe RL* learning scheme is put forth in [16], but with a centralized implementation.

A key promise of designing policies is to alleviate the cyber burden of inverter control. This critical aspect has been largely overlooked by the existing literature. In particular, references

[15] and [16], which are most closely related to this work, update policies continuously and require considerable amounts of data to be communicated in real time. To account for this aspect, the contributions of this work are in two fronts: First, inverter policies are modeled as DNNs that are jointly trained in a quasi-stationary fashion, while feeder constraints are enforced explicitly in a stochastic sense. Second, a carefully designed DNN architecture accommodates application scenarios where inverter rules are driven by local measurements as well as a low-bandwidth control signal broadcast by the utility.

Outline: Section II formulates the task of designing inverter control policies after reviewing an approximate grid model. Section III adopts a stochastic primal-dual algorithm to find the optimal inverter control policies. Section IV puts forth the novel communication-cognizant DNN-based inverter control architecture. The proposed schemes are evaluated using real-world solar generation and load data on the IEEE 13-bus feeder in Section V. Conclusions along with ongoing and future research directions are discussed in Section VI.

Notation: lower- (upper-) case boldface letters denote column 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. GRID MODELING AND PROBLEM FORMULATION

Consider a feeder with $N+1$ buses, including the substation indexed by 0. 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 $q_n = q_n^g - q_n^c$. If vectors $(\mathbf{p}, \mathbf{q})$ collect the power injections at all non-substation buses, they can be decomposed as $\mathbf{p} = \mathbf{p}^g - \mathbf{p}^c$ and $\mathbf{q} = \mathbf{q}^g - \mathbf{q}^c$. We refer to the values of (re)active loads and active solar generation at all non-substation buses as *grid conditions*

$$\mathbf{z} := [(\mathbf{p}^c)^\top \ (\mathbf{q}^c)^\top \ (\mathbf{p}^g)^\top]^\top. \tag{1}$$

Given $\mathbf{z}$, the task of reactive power control by DERs aims at optimally setting $\mathbf{q}^g$ to minimize a feeder-wide objective while complying with network and inverter limitations. Starting with the latter, the reactive power injected by inverter n is limited by a given $\bar{q}_n^g$ due to apparent power limits. Apparent power constraints are local and will be collectively denoted by

$$\mathbf{q}^g \in \mathcal{Q} := \{\mathbf{q} : |q_n^g| \leq \bar{q}_n^g \quad \forall n\}. \tag{2}$$

Regarding feeder constraints, the focus is on confining voltages within the regulation range of $[0.97, 1.03]$ per unit (pu). Albeit voltages are nonlinearly related to power injections, for simplicity we adopt a widely used linearized grid model [17]. According to this model, the vector of voltage magnitudes at all N buses is approximately

$$\mathbf{v} = \mathbf{R}\mathbf{p} + \mathbf{X}\mathbf{q} + v_0\mathbf{1} \tag{3}$$

where v_0 is the substation voltage, while the symmetric positive semidefinite matrices $(\mathbf{R}, \mathbf{X})$ depend on the feeder

and are assumed to be known. If each voltage v_n is to be maintained within $[\underline{v}_n, \overline{v}_n]$, the reactive power injections $\mathbf{q}^g$ should satisfy the network constraints

$$\mathbf{g}(\mathbf{q}^g, \mathbf{z}) := \left[\begin{array}{c} \mathbf{X}\mathbf{q}^g + \mathbf{y} - \overline{\mathbf{v}} \\ -\mathbf{X}\mathbf{q}^g - \mathbf{y} + \underline{\mathbf{v}} \end{array} \right] \leq \mathbf{0} \tag{4}$$

where vector $\mathbf{y} := \mathbf{R}(\mathbf{p}^g - \mathbf{p}^c) - \mathbf{X}\mathbf{q}^c + v_0\mathbf{1}$ depends on $\mathbf{z}$, and vectors $(\underline{\mathbf{v}}, \overline{\mathbf{v}})$ contain the limits $(\underline{v}_n, \overline{v}_n)$ across buses.

According to the same grid model, ohmic losses on lines can be approximated as a convex quadratic function of power injections as $\mathbf{p}^\top\mathbf{R}\mathbf{p} + \mathbf{q}^\top\mathbf{R}\mathbf{q}$; see [17] for details. Upon defining $\mathbf{b} := 2\mathbf{R}\mathbf{q}^c$, the part of ohmic losses that is dependent on the control variable $\mathbf{q}^g$ can be approximated as

$$\ell(\mathbf{q}^g, \mathbf{z}) = (\mathbf{q}^g)^\top\mathbf{R}\mathbf{q}^g - \mathbf{b}^\top\mathbf{q}^g. \tag{5}$$

We henceforth abuse notation and use $\mathbf{q}$ in lieu of $\mathbf{q}^g$. This should not cause any confusion since $\mathbf{q}^c$ has been included in $\mathbf{z}$. DER reactive setpoints $\mathbf{q}$ can be found as the minimizer of

$$\min_{\mathbf{q}\in\mathcal{Q}} \quad \ell(\mathbf{q}, \mathbf{z}) \tag{6}$$
$$\text{s.to} \quad \mathbf{g}(\mathbf{q}, \mathbf{z}) \leq \mathbf{0}.$$

Under the linearized grid model, the approximate OPF task of (6) is a convex quadratic program (QP). Solving (6) can be computationally and communication-wise taxing if $\mathbf{z}$ changes frequently. Moreover, by the time (6) is solved and decisions are downloaded to DERs, grid conditions $\mathbf{z}$ may have changed rendering the computed setpoints obsolete.

To account for the uncertainty in $\mathbf{z}$, one may pursue a stochastic formulation such as [3]

$$\min_{\mathbf{q}\in\mathcal{Q}} \quad \mathbb{E}[\ell(\mathbf{q}, \mathbf{z})] \tag{7}$$
$$\text{s.to} \quad \mathbb{E}[\mathbf{g}(\mathbf{q}, \mathbf{z})] \leq \mathbf{0}$$

where the expectation $\mathbb{E}$ is with respect to $\mathbf{z}$. Nonetheless, the obtained 'one-size-fits-all' $\mathbf{q}$ does not adapt to different $\mathbf{z}$'s.

To come up with DER setpoints that are responsive to grid conditions, we resort to *control policies or rules*, where the reactive power setpoint for each inverter n is captured by a function $\pi_n(\mathbf{w}_n; \boldsymbol{\theta}_n)$ acting upon a control input $\mathbf{w}_n$ and is parameterized by vector $\boldsymbol{\theta}_n$. Ideally, inverter control policies should be driven by the complete $\mathbf{z}$, that is $\mathbf{w}_n = \mathbf{z}$ for all n. Nevertheless, that would entail high communication overhead. If the utility knows the complete $\mathbf{z}$, it might as well solve (6) and communicate the optimal setpoints to inverters. For an inverter control scheme to be communication-cognizant, the inputs $\mathbf{w}_n$ should primarily involve local readings of $\mathbf{z}$, such as (p_n^g, q_n^g, p_n^g), and possibly few remote entries. Regarding the parameter vectors $\boldsymbol{\theta}_n$'s, these may be unique per inverter or share some entries as detailed in Section IV. To capture the aforementioned scenarios, let us abstractly refer to the vector of inverter policies $\pi_n(\mathbf{w}_n; \boldsymbol{\theta}_n)$'s as

$$\mathbf{q}(\mathbf{w}) = \boldsymbol{\pi}(\mathbf{w}; \boldsymbol{\theta}) \tag{8}$$

where $\mathbf{w}$ is the union of $\mathbf{w}_n$'s and $\boldsymbol{\theta}$ the union of $\boldsymbol{\theta}_n$'s.

The control policies for DERs can be found jointly by solving the constrained stochastic minimization

$$P^* := \min_{\boldsymbol{\theta}:\boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta})\in\mathcal{Q}} \mathbb{E}[\ell(\boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta}),\mathbf{z})] \qquad (9)$$
$$\text{s.to} \quad \mathbb{E}[\mathbf{g}(\boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta}),\mathbf{z})] \leq \mathbf{0}$$

over the parameter vector $\boldsymbol{\theta}$. Problem (9) couples policies in two ways. First, for a fixed $\mathbf{z}$, policies are coupled across inverters through the cost and constraint functions since the entries of $\mathbf{q}^g$ appearing in (3) and (5) are now computed via (8). Second, the expectations in (7) and (9) couple system's performance across OPF instances characterized by $\mathbf{z}$.

Local and linear policies of the form $\pi_n(\mathbf{w}_n;\boldsymbol{\theta}_n) = \boldsymbol{\theta}_n^\top \mathbf{w}_n$ have been previously studied for inverter control [5], [18], [19]. Nonetheless, the optimal policies $q_n(\mathbf{w}_n)$ are not necessarily affine in $\mathbf{w}_n$, especially when $\mathbf{w}_n$ is only a partial observation of $\mathbf{z}$. The grand challenge towards scalable inverter control is to design *nonlinear control curves*. In [13], we dealt with by modeling each $q_n(\mathbf{w}_n)$ as a kernel-based support vector machine (SVM), and designing all rules jointly under 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 inputs $\mathbf{w}_n$'s can be challenging. Inspired by their field-changing performance in various engineering tasks, here we propose modeling inverter rules using DNNs, and train the parameters $\boldsymbol{\theta}$ in a data-driven physics-aware fashion.

III. PRIMAL-DUAL DNN LEARNING

Solving (9) is challenging since it is a constrained stochastic minimization over a DNN. To train the inverter policy DNN, we adopt the stochastic primal-dual updates of [20], which are briefly reviewed next. Consider the Lagrangian function of (9)

$$L(\boldsymbol{\theta};\boldsymbol{\lambda}) = \mathbb{E}[\ell(\boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta}),\mathbf{z})] + \boldsymbol{\lambda}^\top \mathbb{E}[\mathbf{g}(\boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta}),\mathbf{z})] \quad (10)$$

where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers corresponding to constraint (9). The dual problem can be posed as

$$D^* = \max_{\boldsymbol{\lambda}\geq\mathbf{0}} \min_{\boldsymbol{\theta}:\boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta})\in\mathcal{Q}} L(\boldsymbol{\theta};\boldsymbol{\lambda}). \qquad (11)$$

Standard duality results predicate that $D^* \leq P^*$. When the primal problem is convex, the previous inequality typically holds with equality. Problem (9) however is non-convex even if (6) is a convex QP, since the DNN mapping $\boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta})$ is generally non-convex in $\boldsymbol{\theta}$. Nonetheless [20] establishes that: *i)* under relatively mild conditions satisfied by (9), and *ii)* if the underlying DNN architecture is rich enough, the duality gap $P^* - D^*$ is sufficiently small. This motivates solving (9) through the primal-dual updates indexed by k [20]

$$\boldsymbol{\theta}^{k+1} = \left[\boldsymbol{\theta}^k - \mu_\theta \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^k;\boldsymbol{\lambda}^k)\right]_{\mathcal{Q}} \qquad (12a)$$
$$\boldsymbol{\lambda}^{k+1} = \left[\boldsymbol{\lambda}^k + \mu_\lambda \nabla_{\boldsymbol{\lambda}} L(\boldsymbol{\theta}^{k+1};\boldsymbol{\lambda}^k)\right]_+ \qquad (12b)$$

where the operator $[\cdot]_{\mathcal{Q}}$ projects $\boldsymbol{\theta}^{k+1}$ such that $\boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta}^{k+1}) \in \mathcal{Q}$ for all $\mathbf{w}$; operator $[\cdot]_+$ ensures $\boldsymbol{\lambda} \geq \mathbf{0}$ at all times; and $(\mu_\theta, \mu_\lambda)$ are positive step sizes. Regarding $[\cdot]_{\mathcal{Q}}$, the DNN output corresponding to q_n^g can be constrained within

$[-\bar{q}_n^g, +\bar{q}_n^g]$ by using $\tanh(\cdot)$ as the output activation function and then scaling by the constant $\bar{q}_n^g$.

The updates in (12) are complicated by the expectation operator. The pdf of $\mathbf{z}$ (and hence $\mathbf{w}$) may not be known beforehand. Even if it is known, propagating that pdf through nonlinear functions such as $\boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta}), \mathbf{z}$ is non-trivial. To deal with this, the primal-dual updates of (12) can be surrogated by their stochastic approximation counterparts. In particular, the utility is assumed to have a set of scenarios $(\mathbf{z}^k, \mathbf{w}^k)$ indexed by $k = 1, \ldots, K$, with which the ensemble averages of (10) are approximated as $\mathbb{E}[\ell(\boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta}),\mathbf{z})] \simeq \frac{1}{K}\sum_{k=1}^K \ell(\boldsymbol{\pi}(\mathbf{w}^k;\boldsymbol{\theta}),\mathbf{z}^k)$. To simplify the updates of (12), the sample averages can be approximated by a single scenario per iteration to yield the *stochastic* primal-dual updates [20]

$$\boldsymbol{\theta}^{k+1} = \left[\boldsymbol{\theta}^k - \mu_\theta\left(\nabla_{\boldsymbol{\theta}}\ell^k - (\nabla_{\boldsymbol{\theta}}\mathbf{g}^k)^\top \boldsymbol{\lambda}^k\right)\right]_{\mathcal{Q}} \qquad (13a)$$
$$\boldsymbol{\lambda}^{k+1} = \left[\boldsymbol{\lambda}^k + \mu_\lambda \mathbf{g}\left(\boldsymbol{\pi}(\mathbf{w}^k;\boldsymbol{\theta}^{k+1}),\mathbf{z}^k\right)\right]_+. \qquad (13b)$$

Here $\nabla_{\boldsymbol{\theta}}\ell^k$ is the gradient of $\ell(\boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta}),\mathbf{z})$ and $\nabla_{\boldsymbol{\theta}}\mathbf{g}^k$ the Jacobian matrix of $\mathbf{g}(\boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta}),\mathbf{z})$, both with respect to $\boldsymbol{\theta}$ and evaluated at $(\mathbf{w}^k, \boldsymbol{\theta}^k, \mathbf{z}^k)$. The updates are known to converge to a stationary point of (9) for sufficiently small step sizes.

For the objective and constraint functions of (4)–(5), the needed sensitivities can be computed as

$$\nabla_{\boldsymbol{\theta}}\ell^k = \left(\nabla_{\boldsymbol{\theta}}\boldsymbol{\pi}(\mathbf{w}^k;\boldsymbol{\theta}^k)\right)^\top \left(2\mathbf{R}\boldsymbol{\pi}(\mathbf{w}^k;\boldsymbol{\theta}^k) - \mathbf{b}^k\right)$$
$$\nabla_{\boldsymbol{\theta}}\mathbf{g}^k = [\mathbf{X} \quad -\mathbf{X}]^\top \nabla_{\boldsymbol{\theta}}\boldsymbol{\pi}(\mathbf{w}^k;\boldsymbol{\theta}^k).$$

Here $\mathbf{b}^k := 2\mathbf{R}(\mathbf{q}^c)^k$ and $\nabla_{\boldsymbol{\theta}}\boldsymbol{\pi}(\mathbf{w}^k;\boldsymbol{\theta}^k)$ is the Jacobian matrix of the DNN output with respect to its weight parameters. The latter can be evaluated using gradient back-propagation across the DNN, a standard tool readily available in all DNN-related software. If the number of available grid scenarios K is relatively small, additional scenarios can be synthesized by applying small perturbations on the available $\mathbf{z}^k$'s. As customary in DNN training, the updates (13) can be iterated over multiple epochs or in mini-batch forms.

It is worth contrasting the DNN input $\mathbf{w}$ and the vector of grid conditions $\mathbf{z}$. Despite some possible overlap, the two vectors are used differently. The former one feeds the DNN to compute the setpoints $\mathbf{q}(\mathbf{w}) = \boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta})$. The latter one is involved in the OPF objective and constraint functions, i.e., it appears in $\mathbf{b}$ for computing $\nabla_{\boldsymbol{\theta}}\ell$ and when evaluating $\mathbf{g}(\boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta}),\mathbf{z})$. While $\mathbf{z}$ should be known to the utility during training to perform the updates of (13), it is not needed during real-time operation. This resonates with the small/big data setup, since a utility has offline access to an extensive smart meter dataset of $\mathbf{z}$'s; yet its control center and each inverter individually are driven by limited real-time data feeds. The updates of (13) apply for inverters DNNs of arbitrary architecture. We next particularize the structure of $\boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta})$ to comply with communication limitations in inverter control.

IV. COMMUNICATION-COGNIZANT DNN ARCHITECTURE

To coordinate inverters on a tight communication budget, our proposed inverter policy DNN $\boldsymbol{\pi}(\mathbf{w};\boldsymbol{\theta})$ comes with the
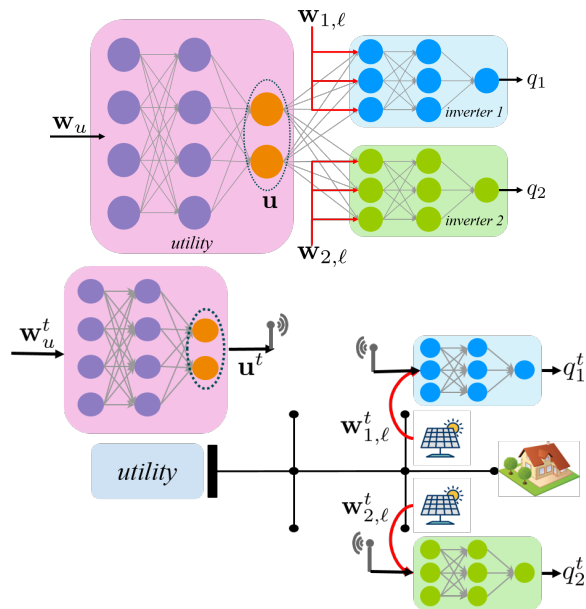
Fig. 1. *Top*: DNN $\boldsymbol{\pi}(\mathbf{w}; \boldsymbol{\theta})$ is organized in one utility sub-NN and inverter sub-NNs, all trained as a single DNN by the utility offline. *Bottom*: During real-time operation, the utility sub-NN uses real-time data to compute and broadcast the control signal, while inverter sub-NNs are run at inverters.

two-tier architecture shown on Figure 1 (top). Its first layers constitute the *utility sub-NN*, while the final layers constitute the *inverter sub-NNs*, one for each inverter. Figure 1 shows only two inverters for simplicity. The utility sub-NN (shown in purple) is fully connected, is driven by input $\mathbf{w}_u$, and outputs control $\mathbf{u}$. Inverter sub-NNs (in blue and green) are disconnected from each other and both fed with the common control $\mathbf{u}$. Each inverter sub-NN is also fed with its own local data $\mathbf{w}_{n,\ell}$. The n-th inverter sub-NN predicts the setpoint q_n.

Inverter policy n can be expressed as $q_n(\mathbf{w}_n) = \pi_n(\mathbf{w}_n; \boldsymbol{\theta}_n)$ where $\mathbf{w}_n = [\mathbf{w}_u^\top \ \mathbf{w}_{n,\ell}^\top]^\top$ and $\boldsymbol{\theta}_n$ collects the DNN parameters for the shared utility sub-NN and inverter sub-NN n. Vectors $\mathbf{w}_{n,\ell}$ may carry local load and solar generation available on bus n. Input $\mathbf{w}_u$ carries information available to the utility control center in real time. Such information can be power flow readings from major distribution lines, transformers, and/or voltage regulators. Vector $\mathbf{w}_u$ may also carry the solar generation from a solar farm or any other DER that is telemetered in real time. Rather than actual grid measurements, vector $\mathbf{w}_u$ may also include predictions the utility can make on grid conditions. For example, that could be the case if the utility uses cameras to monitor cloud coverage as a proxy to solar generation or temperature/humidity readings to load.

During training and given grid scenario $\mathbf{z}^k$, the inputs $\mathbf{w}_u^k$ and $\mathbf{w}_n^k$'s can be: *i)* found readily as partial entries of $\mathbf{z}^k$ (loads and solar generation); *ii)* inferred from $\mathbf{z}^k$ (a line flow can be computed through the power flow equations, or approximated as the sum of all downstream power injections); or *iii)* found through historical data (dataset combining cloud coverage with solar generation). The particular structure of the proposed DNN with individualized inputs and partially

connected layers can be easily implemented by skipping and masking connections, respectively.

Although trained as a whole, the inverter policy DNN $\boldsymbol{\pi}(\mathbf{w}; \boldsymbol{\theta})$ is implemented in parts; see bottom panel of Fig. 1. After training is completed for the upcoming 30- or 60 min period, the weights corresponding to inverter NNs are downloaded to inverters. A unique component of our DNN architecture is the control signal $\mathbf{u}$, which is broadcast from the utility NN to inverter NNs. To save on downlink (utility to inverters) communications, signal $\mathbf{u}$ is designed to be much shorter than $\mathbf{w}_u$. Considering that $\mathbf{u}$ is actually designed along with the operation of inverter sub-NNs through the OPF of (9), this signal carries all the information the utility can provide to coordinate inverters in a condensed form. Its broadcast nature further contributes to communication savings. The steps involved during the training and real-time operation of the proposed DNN are summarized in Algorithm 1.

This DNN architecture can cater to a wide range of communication specifications. If no downlink communication is allowed in real time, the utility sub-NN can be ignored all together and inverter sub-NNs are driven based on local inputs. If downlink bandwidth is abundant, inverter sub-NNs can be dropped and inverter setpoints can be decided by the utility sub-NN in real time. Practical application scenarios are expected to lie somewhere between these two extremes, whence the hybrid architecture of Fig. 1 becomes relevant.

V. NUMERICAL TESTS

The proposed DNN-based inverter control was evaluated on a single-phase version of the IEEE 13-bus feeder. Real-world active load data was extracted for March 1, 2018, on a one-minute resolution from Pecan Street. Solar generation data was also added to buses $\{1, 5, 9, 10, 11, 12\}$, out of which buses $\{9, 12\}$ were equipped with inverters. Load time series were
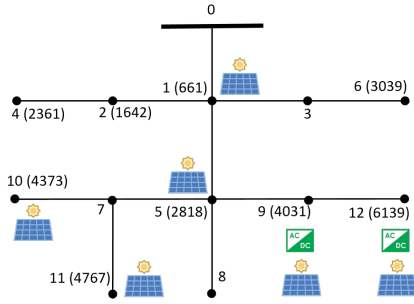

Algorithm 1 Inverter control through DNN-based policies

Training

1: Collect grid scenarios $\{\mathbf{z}^k\}_{k=1}^K$ from smart meter data
2: Collect or calculate DNN inputs $\{\mathbf{w}^k\}_{k=1}^K$
3: Initialize $\boldsymbol{\theta}^0$ and $\boldsymbol{\lambda}^0$
4: **for** all K scenarios and E epochs **do**
5:     Update $\boldsymbol{\theta}$ using (13a)
6:     Update $\boldsymbol{\lambda}$ using (13b)
7: **end for**
8: Download $\boldsymbol{\theta}$ parameters to inverter sub-NNs

Real-time operation

1: **for** $t = 0, 1, \ldots, T,$ **do**
2:     Utility receives $\mathbf{w}_u^t$ from real-time telemetry
3:     Feed $\mathbf{w}_u^t$ to utility sub-NN to compute $\mathbf{u}^t$
4:     Utility broadcasts $\mathbf{u}^t$ to inverters
5:     **for** each inverter n **do**
6:         Inverter n reads $\mathbf{u}^t$ and local data $\mathbf{w}_n^t$
7:         Feed $(\mathbf{u}^t, \mathbf{w}_n^t)$ to inverter sub-NN to decide q_n^t
8:     **end for**
9: **end for**

Fig. 2. The IEEE 13-bus feeder. Numbers in parentheses indicate the house index from the Pecan Street dataset mapped to each bus.

Fig. 3. Training: Average losses under no solar for 12–1 am *(top)* and high solar for 1–2 pm *(second)*. Voltage excursions for 1–2 pm *(third)*. Dual variable for active constraint on bus 11 for 1–2 pm *(bottom)*

scaled so that monthly peaks were 7.5 times the benchmark values. The same ratio was used to scale solar. Reactive loads were added with lagging power factors sampled from a uniform distribution between 0.9 and 1. The utility was assumed to have telemetry $\mathbf{w}_u$ for the active line flows feeding buses $\{2, 3, 7\}$ from their parent buses.

The utility sub-NN was constructed using an input layer of dimension 3 and an output layer $\mathbf{u}$ of dimension 1. Inverter sub-NNs were made up of one input, hidden, and output layers of dimensions 5, 6 and 1, respectively. The local readings $\{p_n, q_n^c\}$ along with $\mathbf{u}$ were fed as inputs to each inverter sub-NN n. Initial values for DNN parameters were uniformly sampled from the range $[-0.1, 0.1]$ and updated using Adam with a learning rate of 0.01. The dual variables were all initialized at 0 and updated with step sizes of 1 that decayed with the square-root of the iteration index [21]. Our approach was contrasted with an optimal policy $\mathbf{q}(\mathbf{w})$ that directly solves (7) without being confined to a DNN parameterization using dual decomposition [21]. As in (7), the optimal policy regulates the average rather than the instantaneous voltages.

We assumed one-hour long control periods. Training scenarios were obtained from the 60 one-minute data observed over the preceding control period. The original grid scenarios were augmented by adding zero-mean additive white Gaussian noise to generate a total of $K = 240$ scenarios. All scenarios were then randomly shuffled. The variance of the additive noise was decided on the basis of training samples observed and was set to 10^{-6} pu for low-solar and 10^{-2} pu for high-solar hours. DNN $\boldsymbol{\pi}(\mathbf{w}; \boldsymbol{\theta})$ was trained using Alg. 1 for 30 epochs.

Figure 3 shows the average losses obtained during training. The losses under our solution were found to be only slightly larger than those attained by the optimal policy. During the low-solar scenario, the base case without any reactive power compensation does not experience any voltage excursions. Therefore, both the optimal policy and our solution focus on decreasing the average losses. On the other hand, when solar generation is high, the basecase experiences high voltage excursions as seen in panel 3. Consequently, the optimal policy and our solution focus on lowering average voltages by withdrawing reactive power at the expense of increased ohmic losses. As demonstrated by the third panel, the proposed scheme attained voltage deviations close to those achieved by
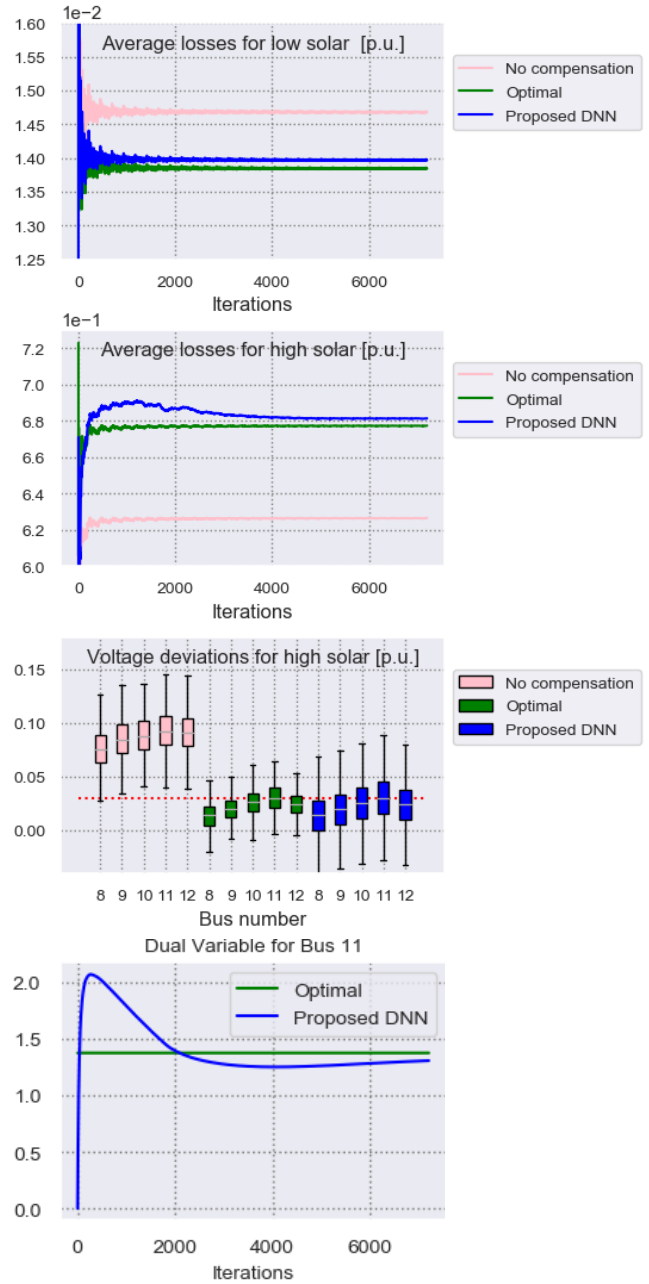
the optimal policy. This near-optimal behavior is also shown in the bottom panel presenting the convergence of dual variables for the active constraint on bus 11 during 1:00–2:00 pm.

The DNNs trained over 12:00–1:00 am and 1:00–2:00 pm were tested on the subsequent hours 1:00–2:00 am and 2:00–3:00 pm, respectively. The results are presented in Fig. 4. The proposed scheme again closely matches the performance of the optimal policy in terms of both minimizing loses and imposing voltage constraints. This is remarkable especially because the optimal policy has access to perfect forecasts and incurs a large real-time communication overhead, while the DNN-based scheme is trained only on historical data and
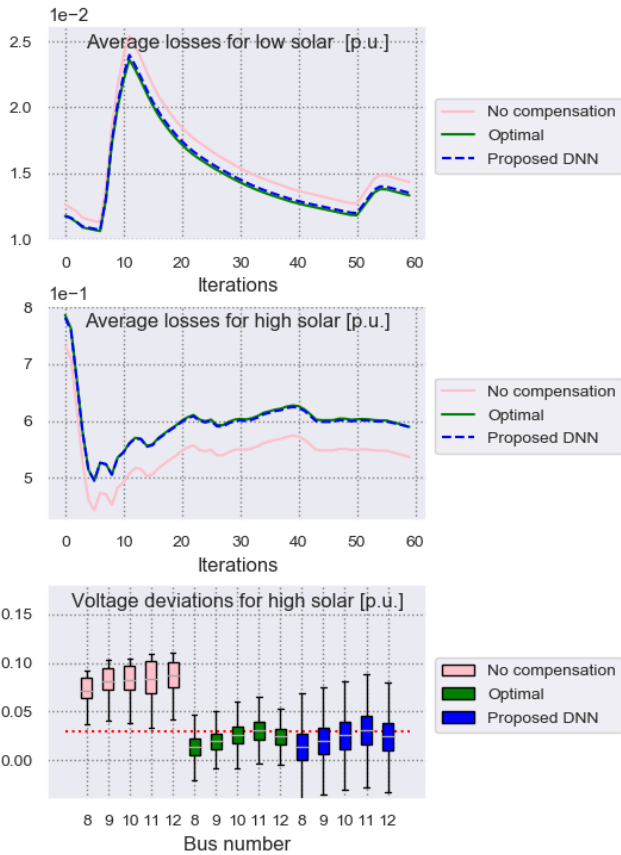
Fig. 4. Testing. Average losses for 1–2 am *(top)* and 2–3 pm *((middle))*; voltage excursions for 2–3 pm *(bottom)*.

requires only 1 data point to be transmitted in real time.

VI. CONCLUSIONS AND ONGOING WORK

This work has introduced nonlinear control policies for inverter setpoints via a novel two-tier communication-cognizant DNN architecture. The DNN consists of a utility sub-NN and inverter sub-NNs, all jointly trained at the utility at the beginning of every control period, while explicitly incorporating average feeder constraints via primal-dual learning. Upon training, the weights of inverter sub-NNs are downloaded to inverters for real-time implementation. Inverter sub-NNs are driven by local inputs and a control signal broadcast by the utility. Depending on communication specifications, the proposed architecture can accommodate from purely local to centralized and hybrid protocols. Tests on real-world data validate this methodology is capable of reducing ohmic losses and enforcing feeder constraints with little communication overhead. Furthermore, the proposed DNN-based policies perform comparably to stochastic approximation-based optimal policies during both training and testing.

These promising results set the foundations for relevant generalizations. We are currently working on the following directions: *d1)* Model-free primal-dual learning of DNNs that does not require explicit knowledge of the feeder topology, parameters, and/or precise loading conditions during training; *d2)* Chance-constraint formulations; *d3)* Quantify the performance

of the proposed DNN-based approach when compared to the optimal policy; *d4)* incorporating exact AC feeder models; and *d5)* testing on larger feeders to demonstrate scalability.

REFERENCES

[1] K. Turitsyn, P. Sulc, S. Backhaus, and M. Chertkov, "Options for control of reactive power by distributed photovoltaic generators," *Proc. IEEE*, vol. 99, no. 6, pp. 1063–1073, Jun. 2011.

[2] 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.

[3] 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, Nov. 2016.

[4] V. Kekatos, L. Zhang, G. B. Giannakis, and R. Baldick, "Fast localized voltage regulation in single-phase distribution grids," in *Proc. IEEE Intl. Conf. on Smart Grid Commun.*, Miami, FL, Oct. 2015, pp. 725–730.

[5] 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.

[6] R. Dobbe, O. Sondermeijer, D. Fridovich-Keil, D. Arnold, D. Callaway, and C. Tomlin, "Towards distributed energy services: Decentralizing optimal power flow with machine learning," *IEEE Trans. Smart Grid*, vol. 11, no. 2, pp. 1296–1306, Mar. 2020.

[7] 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.

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

[9] 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

[10] 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

[11] 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.

[12] 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.

[13] 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.

[14] Q. Yang, G. Wang, A. Sadeghi, G. B. Giannakis, and J. Sun, "Two-timescale voltage control in distribution grids using deep reinforcement learning," *IEEE Trans. Smart Grid*, vol. 11, no. 3, pp. 2313–2323, May 2020.

[15] 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

[16] 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.

[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] 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.

[19] 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.

[20] 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, 2019.

[21] 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.