IEEE Open Access Journal of
Power and Energy

Received 23 July 2021; revised 26 October 2021; accepted 23 December 2021.
Date of publication 4 January 2022; date of current version 25 April 2022.

Digital Object Identifier 10.1109/OAJPE.2022.3140314

A Meta-Learning Approach to the Optimal Power Flow Problem Under Topology Reconfigurations

YEXIANG CHEN[1], SUBHASH LAKSHMINARAYANA[1] (Senior Member, IEEE), CARSTEN MAPLE[2] (Member, IEEE), AND H. VINCENT POOR[3] (Life Fellow, IEEE)

[1]School of Engineering, The University of Warwick, Coventry CV4 7AL, U.K.
[2]Warwick Manufacturing Group, The University of Warwick, Coventry CV4 7AL, U.K.
[3]Department of Electrical and Computer Engineering, Princeton University, Princeton, NJ 08544 USA

CORRESPONDING AUTHOR: S. LAKSHMINARAYANA (subhash.lakshminarayana@warwick.ac.uk)

This work was supported by the PETRAS National Centre of Excellence for IoT Systems Cybersecurity through the U.K. EPSRC under Grant EP/S035362/1, and in part by the C3.ai Digital Transformation Institute and the U.S. National Science Foundation under Grant ECCS-2039716.

ABSTRACT Recently there has been a surge of interest in adopting deep neural networks (DNNs) for solving the optimal power flow (OPF) problem in power systems. Computing optimal generation dispatch decisions using a trained DNN takes significantly less time when compared to conventional optimization solvers. However, a major drawback of existing work is that the machine learning models are trained for a specific system topology. Hence, the DNN predictions are only useful as long as the system topology remains unchanged. Changes to the system topology (initiated by the system operator) would require retraining the DNN, which incurs significant training overhead and requires an extensive amount of training data (corresponding to the new system topology). To overcome this drawback, we propose a DNN-based OPF predictor that is trained using a meta-learning (MTL) approach. The key idea behind this approach is to find a common initialization vector that enables fast training for any system topology. The developed OPF-predictor is validated through simulations using benchmark IEEE bus systems. The results show that the MTL approach achieves significant training speed-ups and requires only a few gradient steps with a few data samples to achieve high OPF prediction accuracy and outperforms other pretraining techniques.

INDEX TERMS Deep neural networks, meta-learning, optimal power flow, topology reconfiguration.

I. INTRODUCTION

THE optimal power flow (OPF) problem involves the computation of minimum cost generation dispatch subject to the power flow equations and the grid's operational constraints (e.g., voltage/power flow limits, etc.). Power grid operators must solve the OPF problem repeatedly several times a day in order to ensure economical operation. The OPF problem under the generalized alternating current (AC) power flow model is non-convex, and solving them using conventional optimization solvers can be computationally expensive. The growing integration of renewable energy and the power demand uncertainty necessitates solving the OPF problem repeatedly at a significantly faster time scale (in the order to seconds) to respond to the changing system states, leading to significant computational challenges [1].

To overcome this challenge, there has been a significant interest in adopting machine learning (ML) techniques to speed up the computation of the OPF problem. The ML models can be trained *offline*, and the trained model can be used *online* to support the computation of the optimal generation dispatch. The main advantage of this approach is that online computations are cheap, and hence, they can speed up OPF computation significantly. ML has been applied in a number of different ways to support OPF computation.

The most straightforward approach is to use ML models (e.g., DNNs) to directly learn the mapping from the load inputs to the OPF outputs. The real-time load demands are fed as inputs to the trained ML model, and the corresponding OPF solution is computed as outputs. This approach was used to solve the direct-current optimal power flow (DC-OPF) problem in [2], in which, the inputs to the DNN are the active power demand at the load buses and the outputs are the active generation power. This approach was shown to provide up to 100 times speed-up as compared to using

conventional optimization solvers. A similar approach was used to solve the AC-OPF problem in [3], in which, the inputs to the DNN are the active/reactive power demand at the load buses and the outputs are the active power generations and voltage magnitudes at the generator buses. This framework was shown to achieve 20 times speed-up as compared to conventional OPF solvers. A similar approach has been used for other applications such as scheduling under outages [4]. During the training stage, the outage schedules are used as inputs to the DNN, and the corresponding OPF costs are obtained as the DNN outputs. This model can effectively assess the impact of a given outage schedule on the OPF solution. Furthermore, ML methods have been used to provide decentralized decision support for distributed energy resources (DERs). For example, [5] designs a local controller by training an ML model using the historical generation and consumption data. The developed model is used for scheduling generation that minimizes the cost of DER control and network loss. In [6], ML methods are used to predict the optimal inverter actions (DER control policy) based on local measurements.

Different from this approach, ML can also be used indirectly to speed up conventional optimization solvers. For example, ML can be used to learn the set of active constraints at optimality; this approach was used to solve the DC-OPF problem in [7]–[9]. Alternatively, ML can also be used to compute the so-called *warm* start points for optimization solvers, an approach that is especially useful to solve the nonlinear AC OPF problem [10], [11]. Compared to these indirect approaches [7]–[11], the direct approach can achieve greater computational speed-up.

Other machine learning techniques have also been adopted for the OPF problem. For instance, [12] proposes a stacked extreme learning machine to speedup the parameter tuning process and reduce the learning complexity. Reference [13] builds a random forest model to calculate a near-optimal OPF solution and to perform post-contingency analysis. Further, [14] compares the performance of OPF solvers developed according to different ML methods (random forest, multi-target decision tree, and extreme learning machine). The results show that ML methods can significantly reduce the OPF computation time with minimal constraint violations and optimality loss.

Recent works have also provided feasibility guarantees, i.e., provide theoretical results to show that the solutions proposed by the ML models satisfy the power grid's operational constraints (e.g., line/voltage limits, etc.). In particular, a *preventive framework* to ensure feasibility for the DC OPF problem was proposed in [15] by calibrating the transmission line capacity limits and the slack bus generation limits to compensate for the inherent approximation errors of DNNs. Similar ideas were extended for the AC OPF problem in [3]. The worst-case guarantees with respect to physical constraint violations for the DNN's OPF solution were derived in [16], [17], and the results were used to reduce the worst-case error. Reference [18] combined DNNs with robust optimization techniques to directly achieve feasible solutions for the security-constrained OPF problem.

Despite the growing research literature on this topic, a major drawback of existing work [2]–[16], [18] is that they are designed for a specific system configuration. As such, they remain effective only as long as the system topology remains fixed. Nevertheless, topology reconfigurations by transmission switching and impedance changes are essential parts of grid operations that can improve the grid's performance from both operational efficiency and reliability point of view [19]–[21]. These measures have gained increasing attention recently. For instance, perturbation of transmission line reactances (using distributed flexible alternating-current transmission systems, D-FACTS devices [22]) is finding increasing applications in power flow control to minimize the transmission power losses [21] and cyber defense [23]–[25]. Similarly, grid operators also perform transmission switching and topology control to ensure economic and reliable system operations [19], [20].

Active topology control poses significant challenges in the use of DNNs for OPF prediction. A DNN trained under a specific system configuration might not be able to provide correct OPF outputs under a different system configuration. This is because the mapping between the load inputs and the OPF outputs will change due to the changes in the system topology. Indeed, our results show that DNNs trained on a specific topology have a poor generalization performance when the system topology changes. Complete retraining with the new system configuration will require significant amounts of training data and time, thus negating the computational speed-up achieved by DNN prediction.

To address these shortcomings, we propose a novel approach in which we train the DNN-based OPF predictor using a meta-learning (MTL) approach. The main idea behind MTL is to a find good initialization point that enables fast retraining for different system configurations. Specifically, we use the so-called model-agnostic MTL approach [26], which finds the initialization point in such a way that a few gradient steps with a few training samples from any system configuration will lead to good prediction performance. This is accomplished by appropriately tuning the loss function of the offline training phase (that finds the initialization point), such that the ML model (DNN in our case) learns internal features that are broadly applicable to the different tasks at hand (i.e., OPF prediction for different variants of the power grid topology), rather than a specific task [26]. Then during the online training phase, these features can be fine tuned to achieve good OPF prediction performance using a few data samples from that topology. Thus the method is well suited to predict OPF solution under planned topology re-configurations. To the best of our knowledge, this work is the first to utilize MTL in a power grid context.

We conduct extensive simulations using benchmark IEEE bus systems. We compare the performance of MTL against several other approaches. They include (i) "Learn from scratch": in which, there is no pretraining, i.e., when the

system is reconfigured, we initialize the DNN weights randomly and train them using the OPF data from the new system reconfiguration. (ii) "Learn from a joint training model": in which, during the offline phase, we train a DNN model from a combined dataset consisting of OPF data from several different topology configurations. Then during the online phase, we initialize the weights of the DNN using this model and fine-tune it using OPF data from the new system configuration. (iii) "Learn from the closet model": in which, during the offline phase, we train several DNN models separately using OPF datasets from different topology configurations (i.e., one DNN for each system configuration). Then, during the online phase, when the topology is reconfigured, we choose the model that achieves the best prediction performance on the new configuration and choose its weight as the initial DNN's weights. The weights are then fine-tuned using OPF data from the new configuration.

We verify the efficacy of the proposed approach by simulations conducted using IEEE bus systems. We generate the OPF data using the MATPOWER simulator and implement the ML models using Pytorch. The results show that the proposed MTL approach can achieve significant training speed-ups and achieve high accuracy in predicting the OPF outputs. For instance, for the IEEE-118 bus system, MTL can achieve greater than 99% OPF generation prediction accuracy for a new system configuration with less than 10 gradient updates and 50 training samples. Furthermore, MTL can achieve a much higher prediction accuracy as compared to complete retraining (i.e., training from scratch), especially in the limited data regime (i.e., when the number of training data samples from a new system configuration are limited). MTL also outperforms the other two pretraining methods in terms of the OPF prediction accuracy and takes significantly less time/storage in the pretraining phase. Thus the method is well suited to predict the OPF solution under planned topology reconfigurations.

We summarize our main contributions in the following:

- To address the shortcoming of existing works that train DNNs under a fixed topology setting and require complete retraining following topology reconfiguration, we propose an MTL approach for computing the OPF solution. Specifically, the MTL approach finds a good initialization point during offline training that enables fast retraining for different system configurations.
- We compare the performance of the MTL approach against several other pretraining methods that are designed to compute the OPF solution following topology reconfigurations. To this end, we perform OPF computation considering several benchmark IEEE bus systems.
- Using simulation results, we quantify the performance gain of the MTL approach as compared to other pretraining methods in terms of the OPF prediction accuracy, feasibility, and computational speed. Our results show that MTL outperforms other pretraining methods on all these metrics, making it suitable for computing OPF under real-world settings that include topology reconfigurations.

The rest of this paper is organized as follows. Section II introduces the power grid model, OPF problem and DNN approach. Section III details the proposed MTL method. Section IV presents the simulation setting. Section V analyses the simulation results and prove the effectiveness of MTL over other pretraining methods. The conclusions are presented in Section VI. Some additional simulation results are included in Appendix.

II. PRELIMINARIES

A. POWER GRID MODEL

We consider a power grid with $\mathcal{N} = \{0, 1, \ldots, N-1\}$ buses, where N is the total number of the buses and $N \geq 2$. Without the loss of generality, we assume bus 0 to be the slack bus whose voltage is set to $1.0\angle 0$ pu. A subset of the buses $\mathcal{G} \subseteq \mathcal{N}$ are equipped with generators. Since the interest of this paper is grid topology reconfigurations, we consider M different grid topologies, where each topology differs with respect to the bus-branch connectivity and transmission line impedances. We assume that the nodes of the power grid always remain connected (among all the considered topologies). We let $\mathcal{L}^{(m)} = \{1, \ldots, L^{(m)}\}$ denote the set of transmission lines under topology $m \in \{1, 2, \ldots, M\}$. Further, we let $\mathbf{Y}^{(m)} = \mathbf{G}^{(m)} + j\mathbf{B}^{(m)}$ denote the bus admittance matrix under topology m, where $\mathbf{G}^{(m)}$ and $\mathbf{B}^{(m)}$ denote conductance and susceptance respectively [27].

Under topology m, let $P_{G_i}^{(m)}$ ($P_{D_i}^{(m)}$) and $Q_{G_i}^{(m)}$ ($Q_{D_i}^{(m)}$) denote the active and reactive power generations (demands) at node $i \in \mathcal{N}$ respectively. The complex voltage at node $i \in \mathcal{N}$ under topology m is denoted by $V_i^{(m)} = |V_i^{(m)}|\angle\theta_i^{(m)}$, where $|V_i^{(m)}|$ is the voltage magnitude and $\theta_i^{(m)}$ is the voltage phase angle. According to the AC power flow model, these quantities are related as

$$P_{G_i}^{(m)} - P_{D_i}^{(m)} = |V_i^{(m)}| \sum_{j\in\mathcal{N}} V_j^{(m)} (G_{i,j}^{(m)} cos(\theta_{i,j}^{(m)}) + B_{i,j}^{(m)} sin(\theta_{i,j}^{(m)})), \quad (1)$$

$$Q_{G_i}^{(m)} - Q_{D_i}^{(m)} = |V_i^{(m)}| \sum_{j\in\mathcal{N}} V_j^{(m)} (G_{i,j}^{(m)} sin(\theta_{i,j}^{(m)}) - B_{i,j}^{(m)} cos(\theta_{i,j}^{(m)})), \quad (2)$$

where $\theta_{i,j}^{(m)} = \theta_i^{(m)} - \theta_j^{(m)}$.

Optimal Power Flow Problem: The OPF problem computes the minimum cost generation dispatch for a given load condition constrained to the power flow equations and power generation/voltage constraints. Mathematically, the OPF problem can be stated as follows:

$$\min_{\substack{P_G^{(m)},\\ Q_G^{(m)}, V^{(m)}}} \sum_{i\in\mathcal{G}} C_i(P_{G_i}^{(m)}) \quad (3)$$

$$s.t.\ (1),\ (2),$$

$$P_{G_i}^{\min} \leq P_{G_i}^{(m)} \leq P_{G_i}^{\max}, \quad \forall i \in \mathcal{G} \quad (4)$$

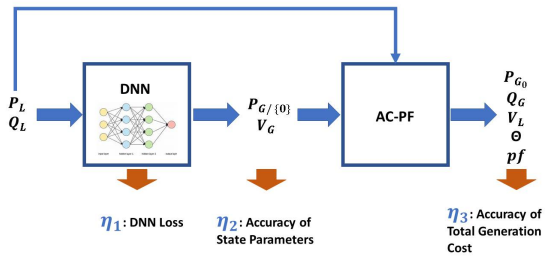


FIGURE 1. The online operation framework of DNN based OPF predictor.

$$Q_{G_i}^{\min} \le Q_{G_i}^{(m)} \le Q_{G_i}^{\max}, \quad \forall i \in \mathcal{G} \tag{5}$$

$$V_i^{\min} \le V_i^{(m)} \le V_i^{\max}, \quad \forall i \in \mathcal{N}, \tag{6}$$

where $C_i(\cdot)$ is the generation cost at bus $i \in \mathcal{G}$. Further, $P_{G_i}^{\max}$ ($P_{G_i}^{\min}$), $Q_{G_i}^{\max}$ ($Q_{G_i}^{\min}$) and $V_i^{\max}$ ($V_i^{\min}$) denote the maximum (minimum) real/reactive power generations and nodal voltage limits at node i respectively.

B. DNN APPROACH FOR THE OPF PROBLEM

We now summarize the approaches proposed by existing works that use DNNs for the OPF problem [2], [3]. Fig. 1 shows an illustration of the overall methodology. The goal of the DNN is to approximate the non-linear mapping between the system load and the OPF solution. Let $\mathbf{h}(\mathbf{x}_k^{(m)}, \mathbf{w})$ denote a parametric function, specifically a DNN under topology m, in our case, that takes the system load as inputs and produces the OPF outputs. Herein, $\mathbf{w}$ denotes the parameters of the DNN. Further, let $\mathcal{T}_m = \{\mathbf{x}_k^{(m)}, \mathbf{y}_k^{(m)}\}_{k=1}^{K_m}$ denote the input-output pair for the OPF problem under configuration m. Herein, K_m denotes the number of training samples and subscript k denotes the training sample's index. For the AC OPF problem, the inputs correspond to the real and reactive power demand at each nodes, i.e., $\mathbf{x}_k^{(m)} = [\mathbf{p}_{D,k}^{(m)}; \mathbf{q}_{D,k}^{(m)}]$, where $\mathbf{p}_{D,k}^{(m)}$ and $\mathbf{q}_{D,k}^{(m)}$ are the vector of real/ reactive power demands, i.e., $\mathbf{p}_{D,k}^{(m)} = [P_{D_i,k}^{(m)}]_{i\in\mathcal{N}}$ (and $\mathbf{q}_{D,k}^{(m)}$ follows a similar definition). The output corresponds to the real power generation dispatch and the generation voltages, i.e., $\mathbf{y}_k^{(m)} = [\mathbf{p}_{G,k}^{(m)}; \mathbf{v}_{G,k}^{(m)}]$ obtained by solving the AC OPF problem. Herein, $\mathbf{p}_{G,k}^{(m)} = [P_{G_i,k}^{(m)}]_{i\in\mathcal{G}\setminus\{0\}}$ consists of a vector of power generation at all buses except the slack bus (note that the generation at the slack bus can be determined by solving the AC power flow problem with the other generations specified) and $\mathbf{v}_{G,k}^{(m)} = [V_{i,k}^{(m)}]_{i\in\mathcal{G}}$. The parameters of the DNN under topology m are trained to minimize the objective function given by

$$J_{\mathcal{T}_m}(\mathbf{w}) = \frac{1}{K_m}\sum_{k=1}^{K_m} ||\mathbf{y}_k^{(m)} - \mathbf{h}(\mathbf{x}_k^{(m)}, \mathbf{w})||^2. \tag{7}$$

This objective function is the mean square error between DNN's predicted value $\mathbf{h}(\mathbf{x}_k^{(m)}, \mathbf{w})$ and the corresponding real value $\mathbf{y}_k^{(m)}$ generated by a traditional OPF solver. Following offline training, the DNN is deployed online to predict the generation outputs for given load inputs. We note that once $[\mathbf{p}_{G,k}^{(m)}; \mathbf{v}_{G,k}^{(m)}]$ are predicted by a trained DNN, the other system parameters (such as the nodal voltages/power injections, etc. at the non-generator buses) can be recovered by solving AC power flow problem as shown in Fig. 1. Note that solving the AC power flow problem is computationally extremely fast as compared to solving the AC OPF problem, and hence, adds only a small computational overhead on the DNN approach [3].

Drawbacks of Existing Work: The main drawback of existing works is that the DNN predictions remain effective only as long as the topology of the system remains fixed. As noted before, topology reconfigurations are increasingly being adopted in power grids to ensure the economic operation and reliability [19]–[21]. While it is certainly possible to retrain the model when the system topology is changed, retraining from scratch will require significant amounts of training data and time. Alternatively, the system operator can train separate DNNs for each system configuration. But this would require a significant amount of computational resources. Moreover, the operator must know all possible topology reconfigurations beforehand, which is not possible, since unforeseen contingencies may arise during power system operations.

III. A META LEARNING APPROACH FOR THE OPF PROBLEM UNDER TOPOLOGY RECONFIGURATION

To overcome these challenges, in this work, we seek to build an ML model for the OPF problem that can be rapidly adapted to a new system configuration. MTL is ideally suited to tackle this problem [26]. MTL is a training methodology that is suited to learn a series of related tasks; when presented with a new and related task, MTL can quickly learn this task from a small amount of training data samples. MTL algorithm consists of two phases, an offline training phase (also called the meta-training phase) and an online training phase (adaptation for the new task). During the offline training phase, MTL finds a set of a good initialization parameters for the series of related tasks. During the online phase, MTL uses the initialization parameter to quickly adapt the model parameters to a new task using a few gradient updates with a few training samples.

A. MTL DESCRIPTION

We now present the details of the proposed MTL approach. As noted in Section I, we consider M different grid topologies. Assume that during the offline training phase, the system operator has access to OPF training data samples from $M^* < M$ topologies. We denote the offline training data set by $\mathcal{T}_{\text{offline training phase}} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{M^*}\}$. During the offline training phase, MTL uses $\mathcal{T}_{\text{offline training phase}}$ to find a set of parameters $\mathbf{w}_{\text{MTL}}$ that minimizes the loss function given by

$$J_{\text{MTL}} = \sum_{m=1}^{M^*} J_{\mathcal{T}_m}(\mathbf{w} - \nabla J_{\mathcal{T}_m}(\mathbf{w})), \tag{8}$$

where $J_{\mathcal{T}_m}$ is defined in (7). The objective function J_{MTL} is the sum of MSE loss for all the topologies in $\mathcal{T}_{\text{offline training phase}}$ following a single-step gradient descent. The MTL parameters are given by $\mathbf{w}_{\text{MTL}} = \arg\min_{\mathbf{w}} J_{\text{MTL}}$. As evident from (8), MTL aims to find an initialization point $\mathbf{w}_{\text{MTL}}$ from which a single gradient update on each topology in $\{1, 2, \ldots, M^*\}$ yields minimal loss on that topology. Since the OPF prediction task under different topologies are related, if we succeed to find a good initialization point for the tasks in $\{1, 2, \ldots, M^*\}$, we can expect this point to be a good initialization point for any topology. Reference [26] proposed a gradient based method to solve the optimization problem (8), which we summarize in Algorithm 1.

Algorithm 1 Offline Training for MTL

Input: Training dataset $\mathcal{T}_{\text{offline training phase}}$, Step sizes α,β
Output: $\mathbf{w}_{\text{MTL}}$: Optimal meta parameter

1: Randomly initialize $\mathbf{w}_{\text{MTL}}$
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_m \in \mathcal{T}_{\text{offline training phase}}$
4: **for all** $\mathcal{T}_m$ **do**
5: Evaluate $\nabla J_{\mathcal{T}_m}(\mathbf{w}_{\text{MTL}})$ using $\mathcal{T}_m$
6: Compute adapted task model parameters with gradient descent: $\mathbf{w}'_m = \mathbf{w}_{\text{MTL}} - \beta \nabla J_{\mathcal{T}_m}(\mathbf{w}_{\text{MTL}})$
7: **end for**
8: Update $\mathbf{w}_{\text{MTL}} \leftarrow \mathbf{w}_{\text{MTL}} - \alpha \nabla \sum_{m=1}^{M} J_{\mathcal{T}_m}(\mathbf{w}'_m)$
9: **end while**
10: Return $\mathbf{w}_{\text{MTL}}$

In Algorithm 1, $\mathbf{w}_{\text{MTL}}$ are the meta-weights (i.e., the initialization weights) for the related tasks, and $\mathbf{w}'_m$ are the task-specific weights for the training topology m (obtained from a single gradient update on $\mathbf{w}_{\text{MTL}}$). The notation $\nabla J_{\mathcal{T}_m}(\mathbf{w})$ denotes the gradient of the loss function (defined in (7) computed using the dataset $\mathcal{T}_m$) with respect and weights $\mathbf{w}$. Finally α and β denote the step sizes for the gradient updates.

During the online training process, assume that the system operator changes the power system topology to a new configuration that does not belong to the dataset in offline training phase. Let $\mathcal{T}^{(new)} \notin \mathcal{T}_{\text{offline training phase}}$ denote the training dataset from the new system configuration. Note that $\mathcal{T}^{(new)}$ may consist of only a few data points $K_{(new)}$ as compared to the offline training data. The objective function of the online training is given by

$$J_{\mathcal{T}^{(new)}}(\mathbf{w}) = \frac{1}{K_{(new)}} \sum_{k=1}^{K_{(new)}} ||\mathbf{y}_k^{(new)} - \mathbf{h}(\mathbf{x}_k^{(new)}, \mathbf{w})||^2. \quad (9)$$

MTL finds the task-specific parameters for this new topology by performing gradient update, which starts from the optimal initialization point $\mathbf{w}_{\text{MTL}}$ obtained in offline training phase.

$$\mathbf{w}_{new} = \mathbf{w}_{\text{MTL}} - \gamma \nabla J_{\mathcal{T}^{(new)}}(\mathbf{w}).$$

The overall procedure for OPF using the MTL approach is presented in Algorithm 2.

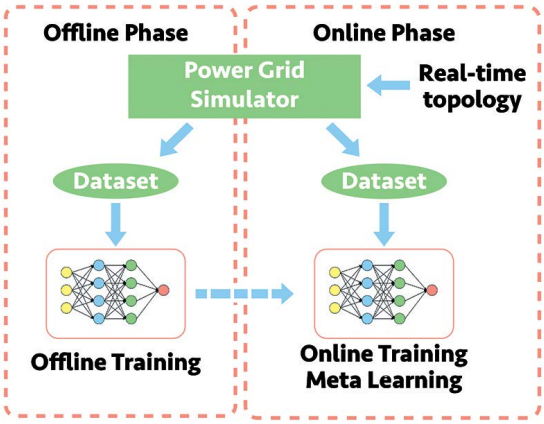


FIGURE 2. **Schematic diagram of MTL implementation for OPF.**

Algorithm 2 MTL Procedure

Input: $\mathbf{w}_{\text{MTL}}$, $\mathcal{T}^{(new)}$, γ
Output: $\mathbf{w}_{new}$: Adapted parameters for new configuration

1: **while** system in operation **do**
2: Change system to new configuration
3: Obtain training samples from the dataset of new configuration $\mathcal{T}^{(new)}$
4: Compute the adapted parameters with gradient descent: $\mathbf{w}_{new} = \mathbf{w}_{\text{MTL}} - \gamma \nabla J_{\mathcal{T}^{(new)}}(\mathbf{w})$
5: **end while**

B. IMPLEMENTATION

A schematic diagram illustrating the proposed MTL implementation is shown in Fig. 2. In the offline phase, the system operator uses a power grid simulator to generate the training data set $\mathcal{T}_{\text{offline training phase}}$. The data is subsequently used to train a DNN as in Algorithm 1. During real-time operation, assume that the system operator plans a topology reconfiguration. During the online training phase, the system operator takes the new system configuration as input to a power grid simulator and generates a few new data samples for the online training phase. Then, the new samples are used to quickly fine-tune the machine learning model as in Algorithm 2. Following retraining, the new model can be used to predict the generator outputs. The online training procedure must be repeated once the system topology is changed.

C. ENSURING FEASIBILITY

The OPF solution predicted by the DNN is feasible when it satisfies the active power generation/ nodal voltage limits, which are specified in (4), (5), (6). In order to ensure the feasibility of DNN proposed solution, we take the following approach proposed in [3], [15]. First, we perform a linear transformation for the active power generation/ nodal voltage magnitudes as follows:

$$P_{G_i}(\rho_i) = \rho_i(P_{G_i}^{\max} - P_{G_i}^{\min}) + P_{G_i}^{\min}, \ \rho_i \in [0, 1], \quad i \in \mathcal{G} \setminus \{0\}, \quad (10)$$

$$V_i(\sigma_i) = \sigma_i(V_i^{\max} - V_i^{\min}) + V_i^{\min},\ \sigma_i \in [0, 1], \quad i \in \mathcal{G}. \tag{11}$$

Note that once we make these transformations, we must have $0 \leq P_{G_i}(\rho_i) \leq 1$, $0 \leq V_i(\sigma_i) \leq 1$. Then, we use the DNN to predict these scaled versions of real power generation and voltages ($P_{G_i}(\rho_i)$, $V_i(\sigma_i)$), rather than predicting P_{G_i} and V_i directly. To this end, at the output layer of the DNN, we use the sigmoid activation function. Recall that the sigmoid function always outputs a number within the range of [0, 1]. Thus, we can guarantee that the prediction of the scaled versions $P_{G_i}(\rho_i)$ and $V_i(\sigma_i)$ predicted by the DNN lie between [0, 1], and consequently, the predictions of P_{G_i} and V_i will lie between their upper and lower limits. Note that without the scaling and the use of the sigmoid function, the DNN prediction cannot be guaranteed to output a feasible solution (i.e., one that lies in between the permissible upper and lower limits).

While the aforementioned transformation ensures the feasibility of the variables directly predicted by the DNN, i.e., $P_{G_i}, i \in \mathcal{G} \setminus \{0\}$ and $V_i, i \in \mathcal{G}$, it does not ensure that feasibility of all the system variables – specifically, those recovered by solving the AC power flow problem (recall Fig. 1). For this reason, we calibrate the voltage constraints while generating the training dataset to avoid such violations [3]. Specifically, in topology m we calibrate the voltage constraints as

$$V_i^{\min} - \lambda \leq V_i^{(m)} \leq V_i^{\max} + \lambda, \quad \forall i \in \mathcal{N}, \tag{12}$$

where λ is a calibration parameter that is set to a small value. This calibration ensures that the DNN is trained to predict voltage magnitudes that lie strictly in the interior of feasible region, and hence mitigates the infeasibility caused by the approximation errors of the DNN. Finally, one can also ensure the feasbility of reactive power generations using a similar procedure. We omit and details here and refer the reader to [3].

IV. SIMULATIONS

In this section, we verify the effectiveness of the proposed MTL approach using simulations and present the results.

A. ALGORITHMS AND METRICS

Under MTL, the offline and online training are performed according to Algorithm 1 and 2. We compare the performance of MTL against three other training methods, namely, ''learn from scratch'' and ''learn from a joint training model'' and ''learn from closet model''.

- In ''Learn from scratch'', there is no pretraining. During the online phase, following topology reconfiguration, a DNN's weights are intialized to random values, and trained using the OPF dataset from the new topology.
- The ''Learn from joint training model'' is described in Algorithm 3. During the offline training phase, a DNN is trained using the dataset $\mathcal{T}_{\text{offline training phase}}$, which combines the training data from topologies $1, \ldots, M^*$. During the online phase, following topology reconfiguration, the DNN's weights are fine-tuned (from the pretrained values) using OPF data from the new topology, similar to the MTL online training phase.
- The ''learn from closet model'' is described in Algorithms 5 and 6. During the offline training phase, we train a separate DNN for each topology $1, \ldots, M^*$. During the online phase, we choose the DNN that achieves the best prediction performance on the new topology at hand (step 4 of Algorithm 6). Then, we fine-tune its weights using OPF data from the new topology.

We henceforth refer to ''Learn from joint training model'' and ''Learn from the closest model'' as ''Pretrain1'' and ''Pretrain2'' respectively.

Algorithm 3 Offline Training for pretrain1

Input: $\mathcal{T}_{\text{offline training phase}}$, α

Output: $\mathbf{w}_{\text{pretrain1}}$: The initial parameters (model) that developed based on joint training

1: **while** not done **do**
2: Update $\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla J_{\mathcal{T}_{\text{offline training phase}}}(\mathbf{w})$
3: **end while**

Algorithm 4 Offline Training for pretrain2

Input: $\mathcal{T}_{\text{offline training phase}}$, α

Output: $W_{pretrain2} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{M^*}\}$: The set of parameters (model) for each task in offline training phase $\mathcal{T}_{\text{offline training phase}}$

1: **for all** $\mathcal{T}^{(m)} \in \mathcal{T}_{\text{offline training phase}}$ **do**
2: **while** not done **do**
3: Update $\mathbf{w}_m \leftarrow \mathbf{w}_m - \alpha \nabla J_{\mathcal{T}_m}(\mathbf{w}_m)$
4: **end while**
$W_{pretrain2} \overset{append}{\Leftarrow} \mathbf{w}_m$
5: **end for**

Algorithm 5 Online Training for pretrain2

Input: $W_{pretrain2}$, $\mathcal{T}^{(new)}$,γ

Output: $\mathbf{w}_{new}$: Adapted parameters for new configuration

1: **while** system in operation **do**
2: Change system to new configuration
3: Obtain training samples from the dataset of new configuration $\mathcal{T}^{(new)}$
4: Find the model that performs best on new task: $\mathbf{w}_{\text{best}} = \arg\min_m J_{\mathcal{T}^{(new)}}(\mathbf{w}_m)$
5: Compute the adapted parameters with gradient descent: $\mathbf{w}_{new} = \mathbf{w}_{\text{best}} - \gamma \nabla J_{\mathcal{T}^{(new)}}(\mathbf{w})$
6: **end while**

The online operation framework of the two-step DNN based OPF solver is presented in Fig 1 (used for the testing

data). In the first step, given the active and reactive power demand at the load buses, the trained DNN predicts the active power generations (except that on the slack bus) and the voltage magnitudes at the generator buses. Then, all other system state parameters (e.g. $P_{\mathcal{G}_0}, Q_{\mathcal{G}}, V_L, \theta$ and branch power flow pf) can be reconstructed by solving simple AC power flow equations.

The performance of the DNN based OPF solver is assessed by three metrics. The first metric η_1 is the DNN validation loss, which is defined in (7). The second metric η_2 is the accuracy of the state parameters, defined in (13), where $2|\mathcal{G}|-1$ is the dimension of DNN output, $\hat{y}_{k,d}^{(m)}$ is the predicted state parameter and $y_{k,d}^{(m)}$ is the corresponding real value. The total generation cost is defined as $cost = \sum_{i\in\mathcal{G}} C_i(P_{G_i}^{(m)})$, and introduced in (3). The third metric η_3 is defined in (14), where $\hat{cost}_k^{(m)}$ is the predicted total generation cost and $cost_k^{(m)}$ is the corresponding real value.

$$\eta_2 = 1 - \frac{1}{K^{(m)}} \sum_{k=1}^{K^{(m)}} \frac{1}{2|\mathcal{G}|-1} \sum_{d=1}^{2|\mathcal{G}|-1} \left| \frac{\hat{y}_{k,d}^{(m)} - y_{k,d}^{(m)}}{y_{k,d}^{(m)}} \right|, \quad (13)$$

$$\eta_3 = 1 - \frac{1}{K^{(m)}} \sum_{k=1}^{K^{(m)}} \left| \frac{\hat{cost}_k^{(m)} - cost_k^{(m)}}{cost_k^{(m)}} \right|, \quad (14)$$

B. DATA CREATION AND DNN SETTINGS

The power system models are based on MATPOWER's test cases [28]. The training and testing data are generated using MATPOWER's AC OPF solver (specifically, we use MATPOWER's interior point solver). We test the algorithms using the IEEE-14, 30 and 118 bus systems. During the data generation phase, we create different power grid topologies by randomly disconnecting a subset of transmission lines (e.g., each line is disconnected with a probability of 0.01) and adding a random perturbation to the line reactance values (subject to a maximum and minimum reactance limit). Some of these topologies may not produce a feasible OPF solution, e.g., if too many transmission lines are disconnected at once, there may not be a feasible solution to the OPF problem that can satisfy the load demand in that topology. Thus, we exclude those topologies from the dataset, since a grid operator will not change the system to those configurations (thus, they dont represent real-world topologies). We keep generating different topologies in the aforementioned manner until we can find sufficient number of ones that have resulted in a feasible OPF solution. For instance, in the 118−bus system, we generated 100 topologies with a feasible OPF solution.

For each bus system, $M = 100$ different grid topologies are generated. For each topology, we create a set of 1000 data points, where each data point corresponds to a different load value obtained by adding a random load perturbation to the base values (that are obtained by the MATPOWER simulator). The maximum load perturbation is restricted to 70% of the original values. We consider the quadratic OPF cost, and use the default generation cost values in MATPOWER.

TABLE 1. The neural network setting for each test case.

Case	Neurons of input layer	Number of hidden layer	Neurons per hidden layer	Neurons of output layer
Case14	22	3	64/32/16	9
Case30	40	3	128/64/32	11
Case118	198	3	256/128/64	107

Changes to the system topology will lead to changes in the power flows, leading to a different OPF solution. In our simulations, we allocate $M^* = 70$ tasks to the offline training phase, and the rest 30 tasks (denoted as new tasks) to the online training phase.

We implement the neural network model and the MTL training based on PyTorch framework. We use the ReLu activation function at the hidden layers, and the sigmoid activation function at the output layer. The size of the input and output layers are chosen to be consistent with the size of the dataset. In our case, the input to the DNN is a vector containing the active and reactive power demand at the load buses. For instance, in the IEEE-118 bus system, the size of the input vector is 198 (corresponding to the active and reactive power demands of the 99 load buses), and the size of the output vector is 107 (corresponding to the $2|\mathcal{G}| - 1$ generator buses). Thus, the number of neurons at the input and output layers are 198 and 107 respectively. For the hidden layers, we vary the number of neurons proportional to the size of the input/output layers. In Table 2, we present the prediction accuracy (η_2) for different number of neurons in the hidden layers considering the IEEE-118 bus system. The setting labelled ''Ref'' provides the highest accuracy, which is the DNN setting we use in the rest of the paper. Similarly, for each test system, we vary the size of the hidden layers and choose the setting that gives the best accuracy results. The settings for each layer under different bus systems used in our simulations are enlisted in Table 1.

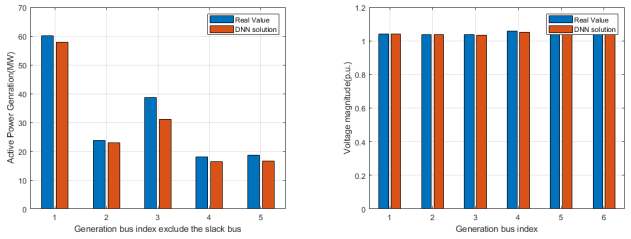
In the offline training process, for each pretraining method, we use the ''Adam'' optimizer with a learning rate of 0.001 and use 1000 training epochs. The L2 regularization is applied to prevent over-fitting, and weight decay is 0.001. For the online training phase, unless specified otherwise, we use 50 training samples during for fine-tuning the weights. Further, we use the stochastic gradient descent (SGD) optimizier with a learning rate is 0.1, and the weight decay is 0.001.

V. SIMULATION RESULTS

The simulation results are presented in Fig. 4,8,9 and Tables 3, 4. For brevity, we only present the results from the IEEE 118 bus system in Fig. 4. The results from the IEEE-14 and 30 bus systems are relegated to the Appendix. The results in all the bus systems follow a similar trend.

TABLE 2. Prediction performance with different neurons in the hidden layers. This is an example based on IEEE-118 bus system.

	Hidden Layers setting	Accuracy (η_2)
Ref	256/128/64	0.9726
Modify Hidden Layer 1	300/128/64	0.9378
	200/128/64	0.9720
	128/128/64	0.9687
Modify Hidden Layer 2	256/200/64	0.9676
	256/100/64	0.9673
	256/64/64	0.9688
Modify Hidden Layer 3	256/128/100	0.9693
	256/128/50	0.9617
	256/128/32	0.9723



(a) Active Power (b) Voltage Magnitude

FIGURE 3. Comparison of real and predicted state parameters in IEEE-30 bus system.

A. COMPARISON OF ACCURACY, FEASIBILITY AND COMPUTATIONAL SPEED DURING ONLINE TRAINING

Fig. 4 and Table 3 present the accuracy results based on the different metrics defined in Section IV. It can be observed that MTL achieves a very high prediction accuracy of over 97% (η_2) and over 99% (η_3) with less than 10 training epochs. This shows that MTL can rapidly adapt to the new system configuration starting from the initialization point $\mathbf{w}_{\text{MTL}}$. In contrast, training from scratch from a random initialization takes a significantly greater number of gradient updates. For the purpose of illustration, we choose one particular system topology from the testing phase and present the results of the true value and the prediction of P_{G_i} and V_i in Fig. 3 for the IEEE-30 bus system, in which we can observe a close match between the two quantities.

Furthermore, MTL also achieves the highest accuracy as compared to the other pretraining methods (Pretrain 1 and 2) and lower loss. More importantly, we also observe that online training with a very number of data samples (i.e., 50 OPF data samples from the new topology in our case) does not significantly improve the performance of other pretraining methods as observed in Table 3 (sometimes, we also observed that for other pretraining methods, online training with only a few data samples may result in worse performance due to over-fitting). Thus, with the other pretraining methods, the accuracy is limited to the performance achieved during the offline training phase.

From Fig. 4 and Table 3, we observe that for MTL, most of the performance improvement occurs within the first few epochs. Thus, MTL is suitable for online training with a very few data samples and a very few training epochs.

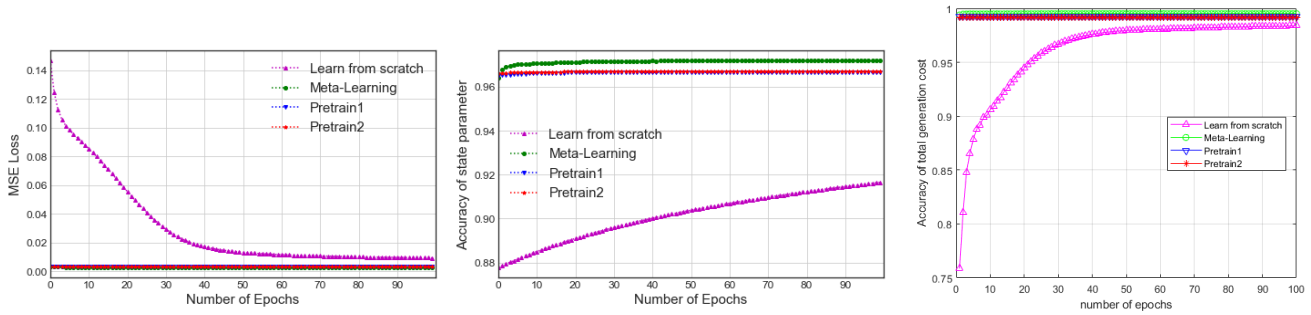
We also present the results for feasibility of the predicted OPF solution in Table 4. The feasibility rate is calculated as $fr = \frac{n_f}{n_t}$, where n_f denotes the number of testing sample that achieves feasible solution, and n_t denotes the total number of testing samples. The results show that the adjustments made to the training process proposed in Section II is able to ensure that MTL achieves very high feasibility rate.

TABLE 3. The online training performance of each pretrain method.

Metric	Method	Epoch 0	Epoch 1	Epoch 10	Epoch100
η_1	MTL	0.0105	0.0040	0.0030	0.0028
	pretrian1	0.0051	0.0051	0.0051	0.0051
	pretrian2	0.0059	0.0060	0.0060	0.0059
η_2	MTL	0.9372	0.9642	0.9707	0.9720
	pretrian1	0.9598	0.9597	0.9597	0.9598
	pretrian2	0.9567	0.9564	0.9565	0.9567
η_3	MTL	0.9886	0.9937	0.9948	0.9950
	pretrian1	0.9925	0.9925	0.9925	0.9925
	pretrian2	0.9915	0.9914	0.9915	0.9915

TABLE 4. Feasibility rate after 100 epochs for each method and test case.

Feasibility / Case / Method	14-bus	30-bus	118-bus
Learn from scratch	0.978	0.989	0.991
MTL	0.998	0.994	0.994
pretrain1	0.989	0.993	0.994
pretrain2	0.989	0.993	0.994



(a) η_1: DNN Loss (b) η_2: Accuracy of State Parameter (c) η_3: Accuracy of Total Generation Cost

FIGURE 4. Visualization of online training progress based on 50 training samples from the new task. Comparison of MTL with other benchmarks using the different metrics for IEEE-118 bus system.

TABLE 5. Computational time for the pretraining methods during the offline training phase.

Pretraining method	14-bus	30-bus	118-bus
MTL	3min 55sec	3min 40sec	9min 6sec
pretrain1	11min 34sec	13min 29sec	104min 31sec
pretrain2	24min 35sec	28min 38sec	151min 16sec

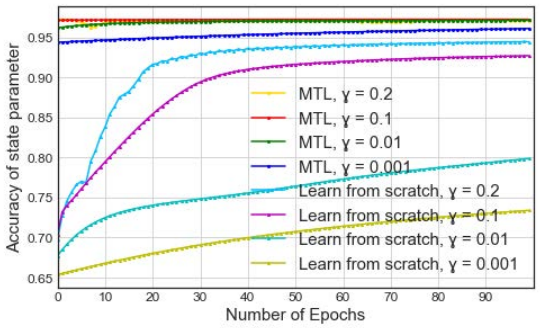


FIGURE 5. MTL/ Learn from scratch, online training performance under different learning rates γ.

B. COMPUTATIONAL TIME FOR THE OFFLINE TRAINING PHASE

Besides the advantages of MTL in terms of accuracy, another advantage is its ability to quickly produce an an initialization model (i.e., the offline training). In Table 5, we enlist the time required to produce the intialization model of MTL and other pretraining methods for different bus systems. It can be observed that MTL takes significantly less time than the other pretraining methods. Moreover, as compared to the Pretrain2 method, which requires a separate DNN to be trained and stored for each power grid topology, MTL requires a single DNN model to be stored. Thus, MTL also significantly reduces the storage burden in comparison to the Pretrain2 method.

C. MTL PERFORMANCE FOR DIFFERENT OFFLINE/ONLINE TRAINING PARAMETERS

We investigate the performance of MTL as a function of the online/offline training parameters. To this end, first, we test the online training performance of MTL/ learn from scratch under different learning rates γ. The result of the IEEE-118 bus system is presented in Fig. 5. Increasing the learning rate γ can accelerate the speed of online training. However, it is not desirable to set a very high learning rate since it may risk oscillations around the minimum (as in gradient update algorithms). For instance, in the result presented in Fig 5, we observe that the learning speed and the accuracy of MTL is enhanced as we increase γ from 0.001 to 0.1. However, when γ is increased beyond this value (for instance $\gamma = 0.2$), the accuracy of the online learning starts to decrease. For each test system, we similarly determine the optimal learning rate by gradually adjusting the value of γ.

Secondly, we investigate the prediction accuracy (measured according to the metric η_2) as a function of the number of training samples used in the online training progress and present the results in Fig. 6. We observe that MTL achieves good accuracy by fine-tuning with only $50 - 100$ online training samples. Increasing the number of online training samples to 700 achieves a negligible improvement in the accuracy. This implies that MTL is good for fine tuning with a very few number of data samples, making it particularly attractive for online training. Note that despite using only a few data samples for during the online ''training'' process of MTL, we have provided significant number of data samples during the ''testing'' phase to ensure sufficient averaging and that the results we present are unbiased. For instance, the result in Fig. 6 is computed based on 300 data samples during the testing phase.

Thirdly, we investigate the MTL prediction accuracy as a function of the number of topologies used in the offline training phase $\mathcal{T}_{\text{offline training phase}}$. The results plotted in Fig. 7 indicate that the prediction accuracy goes down when the number of topologies used in the offline training process is reduced. This indicates that a sufficient number of toplogies

TABLE 6. The online training performance base on operation time.

Case	Epoch	Online training time(ms)	OPF computation time (ms)		Speed Up
			Online prediction time (DNN+PF)	MATPOWER OPF solver time	
14	1	17.1957	0.1821 + 1.4636	20.1868	×13
	10	180.1005			
	100	1928.6110			
30	1	23.7454	0.2193 + 1.6670	40.8395	×22
	10	236.3109			
	100	2327.0433			
118	1	130.3734	0.2353 + 2.6613	52.6581	×19
	10	1209.9692			
	100	12934.5649			

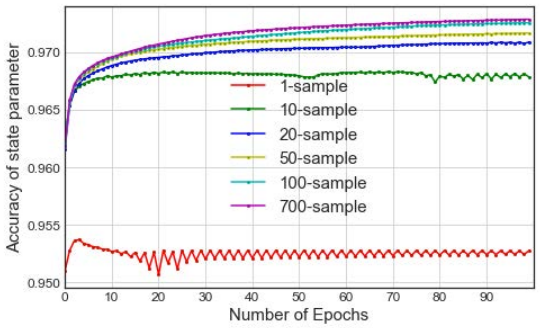


FIGURE 6. Test case under IEEE-118 bus system, the online training performance of MTL based on {1, 10, 20, 50, 100, 700} samples with 70 tasks in offline training phase. The assessment is based on η_2: accuracy of state parameter. Each model is updated according to 'SGD' optimization.The learning rate is 0.1 and weight decay is 0.001.

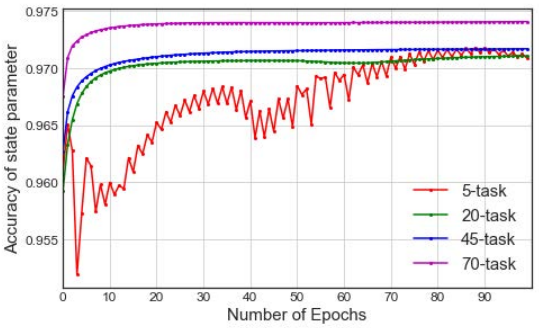


FIGURE 7. Test case under IEEE-118 bus system, with {5, 20, 45, 70} tasks in offline training phase and training based on 50 samples from new task. The assessment is based on η_2: accuracy of state parameter. Each model is updated according to 'SGD' optimization.The learning rate is 0.1 and weight decay is 0.001.

are required in the offline training phase to develop an efficient MTL model.

D. COMPUTATIONAL GAIN COMPARED TO THE TRADITIONAL OPF SOLVER

We further test the computational time for MTL's online training and prediction time (following the retraining) and compare it with the traditional MATPOWER-based OPF solver in Table 6. We summarize the observations in the following.

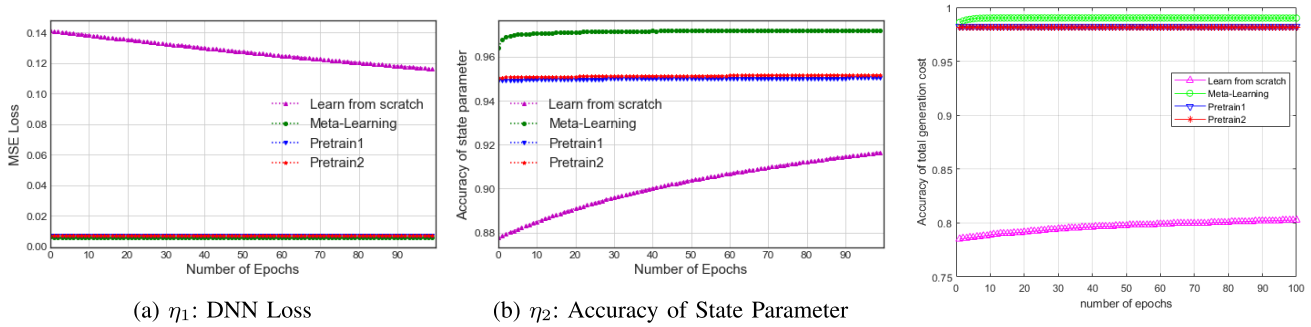
1) ONLINE TRAINING TIME

It can be observed from Table 6 that under the MTL approach, the DNN can be retrained quickly to achieve high prediction accuracy. In particular, recall that MTL's online training achieves a very high prediction accuracy within 10 retraining epochs. For the IEEE-118 bus system, the computation time for the online training (corresponding to 10 epochs) is only 1.2 seconds. Thus, the MTL approach will be scalable to large OPF systems.

2) ONLINE PREDICTION TIME

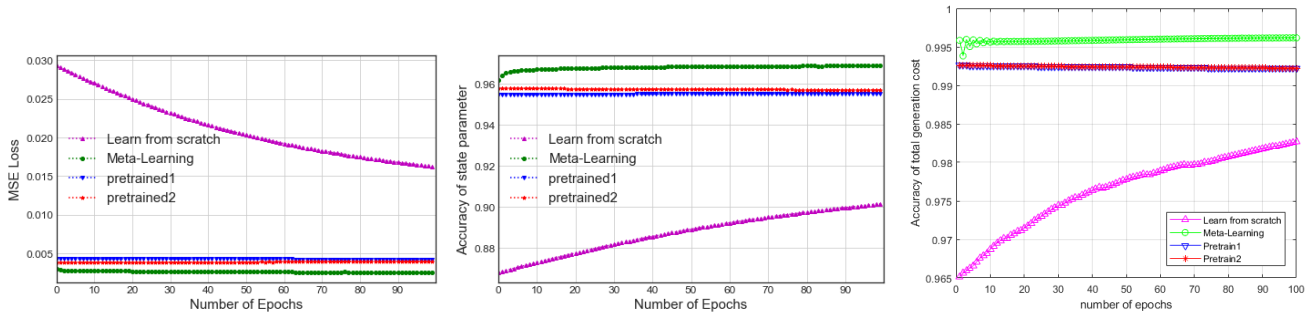
The online prediction phase consists of two steps: (1) DNN prediction (2) post-processing to ensure feasibility (as illustrated in Fig. 1 of the paper). We present the computational time for both these operations in Table 6. We compare it with the time required by the traditional MATPOWER-based OPF solver. The results show that the proposed approach can provide significant speed up in comparison to the traditional solver. For instance, for the IEEE-118 bus system, we can achieve a speed-up of 19 times (for every computation of the OPF).

Finally, note that the online training operation is an additional computation burden incurred under the MTL approach (that is not required by the traditional OPF solver). For the



(a) η_1: DNN Loss (b) η_2: Accuracy of State Parameter (c) η_3: Accuracy of Total Generation Cost

FIGURE 8. Visualization of online training progress based on 50 training samples from the new task. Comparison of MTL with other benchmarks using the different metrics for IEEE-14 bus system. Learning rate = 0.001, Weight decay = 0.001.



(a) η_1: DNN Loss (b) η_2: Accuracy of State Parameter (c) η_3: Accuracy of Total Generation Cost

FIGURE 9. Visualization of online training progress based on 50 training samples from the new task. Comparison of MTL with other benchmarks using the different metrics for IEEE-30 bus system. Learning rate = 0.01, Weight decay = 0.001.

IEEE-118 bus system, we observe that the online training time ($\approx$ 1.2 s) for MTL is approximately 23 times that of traditional OPF solver ($\approx$ 52 ms). From this observation, we can conclude that MTL will be useful for a power system operator if the system topology remains unchanged for at-least 23 OPF computations. If the system topology is changed faster than this rate, then the computational burden of MTL is greater than that of using the traditional OPF solver. However, in most practical systems, this is reasonable, since changes in the load/renewable energy fluctuations occur at a much faster rate compared to the rate of topology reconfigurations. Thus, MTL is suitable in practical power system operation scenarios.

VI. CONCLUSION

In this work, we have proposed a DNN based approach to the OPF problem that is trained using a novel MTL approach. The proposed approach is particularly relevant for computing OPF generation dispatch decisions under power grid topology reconfigurations. The MTL approach finds good initialization points from which the DNNs can be quickly trained to produce accurate predictions for different system configurations. Simulation results show that the proposed approach can significantly enhance the training speed and achieve better prediction accuracy as well as feasible results compared to several other pretraining methods. To the best of our knowledge, this work is the first to adopt an MTL approach in a power grid context.

APPENDIX: SIMULATION RESULTS FOR IEEE-14 AND 30 BUS SYSTEMS

See Figs. 8 and 9.

REFERENCES

[1] Y. Tang, K. Dvijotham, and S. Low, "Real-time optimal power flow," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2963–2973, Nov. 2017.
[2] X. Pan, T. Zhao, and M. Chen, "DeepOPF: Deep neural network for DC optimal power flow," in *Proc. IEEE Int. Conf. Commun., Control, Comput. Technol. Smart Grids (SmartGridComm)*, Oct. 2019, pp. 1–6.
[3] A. S. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," in *Proc. IEEE Int. Conf. Commun., Control, Comput. Technol. Smart Grids (SmartGridComm)*, Nov. 2020, pp. 1–6.
[4] G. Dalal, E. Gilboa, S. Mannor, and L. Wehenkel, "Chance-constrained outage scheduling using a machine learning proxy," *IEEE Trans. Power Syst.*, vol. 34, no. 4, pp. 2528–2540, Jul. 2019.
[5] 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.
[6] R. Dobbe, O. Sondermeijer, D. Fridovich-Keil, D. Arnold, D. Callaway, and C. Tomlin, "Toward distributed energy services: Decentralizing optimal power flow with machine learning," *IEEE Trans. Smart Grid*, vol. 11, no. 2, pp. 1296–1306, Mar. 2020.
[7] Y. Ng, S. Misra, L. A. Roald, and S. Backhaus, "Statistical learning for DC optimal power flow," in *Proc. Power Syst. Comput. Conf. (PSCC)*, Jun. 2018, pp. 1–7.
[8] D. Deka and S. Misra, "Learning for DC-OPF: Classifying active sets using neural nets," in *Proc. IEEE Milan PowerTech*, Jun. 2019, pp. 1–6.
[9] K. Baker and A. Bernstein, "Joint chance constraints reduction through learning in active distribution networks," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Nov. 2018, pp. 922–926.
[10] K. Baker, "Learning warm-start points for AC optimal power flow," in *Proc. IEEE 29th Int. Workshop Mach. Learn. for Signal Process. (MLSP)*, Oct. 2019, pp. 1–6.

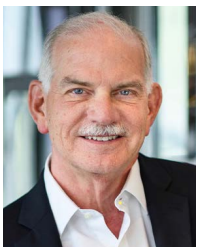
[11] M. Jamei, L. Mones, A. Robson, L. White, J. Requeima, and C. Ududec, "Meta-optimization of optimal power flow," in *Proc. ICML, Climate Change, How Can AI Help Workshop*, 2019, pp. 1–3.
[12] X. Lei, Z. Yang, J. Yu, J. Zhao, Q. Gao, and H. Yu, "Data-driven optimal power flow: A physics-informed machine learning approach," *IEEE Trans. Power Syst.*, vol. 36, no. 1, pp. 346–354, Jan. 2021.
[13] J. Rahman, C. Feng, and J. Zhang, "Machine learning-aided security constrained optimal power flow," in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Aug. 2020, pp. 1–5.
[14] J. Rahman, C. Feng, and J. Zhang, "A learning-augmented approach for AC optimal power flow," *Int. J. Electr. Power Energy Syst.*, vol. 130, Sep. 2021, Art. no. 106908.
[15] X. Pan, T. Zhao, M. Chen, and S. Zhang, "DeepOPF: A deep neural network approach for security-constrained DC optimal power flow," *IEEE Trans. Power Syst.*, vol. 36, no. 3, pp. 1725–1735, May 2021.
[16] A. Venzke, G. Qu, S. Low, and S. Chatzivasileiadis, "Learning optimal power flow: worst-case guarantees for neural networks," in *Proc. IEEE Int. Conf. Commun., Control, Comput. Technol. Smart Grids (SmartGridComm)*, Nov. 2020, pp. 1–7.
[17] A. Venzke and S. Chatzivasileiadis, "Verification of neural network behaviour: Formal guarantees for power system applications," *IEEE Trans. Smart Grid*, vol. 12, no. 1, pp. 383–397, Jan. 2021.
[18] A. Velloso and P. Van Hentenryck, "Combining deep learning and optimization for preventive security-constrained DC optimal power flow," *IEEE Trans. Power Syst.*, vol. 36, no. 4, pp. 3618–3628, Jul. 2021.
[19] K. W. Hedman, S. S. Oren, and R. P. O'Neill, "A review of transmission switching and network topology optimization," in *Proc. IEEE Power Energy Soc. Gen. Meeting*, Jul. 2011, pp. 1–7.
[20] A. S. Korad and K. W. Hedman, "Robust corrective topology control for system reliability," *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 4042–4051, Nov. 2013.
[21] K. M. Rogers and T. J. Overbye, "Some applications of distributed flexible AC transmission system (D-FACTS) devices in power systems," in *Proc. 40th North Amer. Power Symp.*, Sep. 2008, pp. 1–8.
[22] D. Divan and H. Johal, "Distributed FACTS—A new concept for realizing grid power flow control," *IEEE Trans. Power Electron.*, vol. 22, no. 6, pp. 2253–2260, Nov. 2007.
[23] S. Lakshminarayana and D. K. Y. Yau, "Cost-benefit analysis of moving-target defense in power grids," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2018, pp. 139–150.
[24] C. Liu, J. Wu, C. Long, and D. Kundur, "Reactance perturbation for detecting and identifying FDI attacks in power system state estimation," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 4, pp. 763–776, Aug. 2018.
[25] S. Lakshminarayana, E. V. Belmega, and H. V. Poor, "Moving-target defense against cyber-physical attacks in power grids via game theory," *IEEE Trans. Smart Grid*, vol. 12, no. 6, pp. 5244–5257, Nov. 2021.
[26] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.
[27] A. Wood and B. Wollenberg, *Power Generation, Operation, and Control*. Hoboken, NJ, USA: Wiley, 1996.
[28] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, Feb. 2011.

YEXIANG CHEN received the B.S. degree in electrical engineering from Hohai University, Nanjing, China, and the University of Strathclyde, Glasgow, U.K., in 2017, and the M.S. degree in electrical engineering from the University of Strathclyde, in 2019. He is currently pursuing the Ph.D. degree in electrical engineering with The University of Warwick, U.K. His research interests include power systems, cyber-physical system security, and machine learning.

SUBHASH LAKSHMINARAYANA (Senior Member, IEEE) received the B.S. degree from Bangalore University, India, the M.S. degree in electrical and computer engineering from The Ohio State University in 2009, and the Ph.D. degree from the Alcatel Lucent Chair on Flexible Radio and the Department of Telecommunications, SUPELEC, France, in 2012.

He is currently an Associate Professor at the School of Engineering, The University of Warwick, U.K. Previously, he worked as a Researcher at the Advanced Digital Sciences Center (ADSC), Singapore, from 2015 to 2018. He was a Joint Post-Doctoral Researcher at Princeton University and the Singapore University of Technology and Design (SUTD) from 2013- to 2015. His research interests include cyber-physical system security (power grids and urban transportation) and wireless communications. His works have been selected among the best conference papers on integration of renewable and intermittent resources at the IEEE PESGM—2015 Conference and the "Best 50 Papers" of IEEE GLOBECOM 2014 Conference. He regularly serves in the technical program committees for IEEE conferences. He serves as an Associate Editor for the *IET Smart Grid* journal and *Frontiers in Communications and Networks* journal (Smart Grid Communications Section).

CARSTEN MAPLE (Member, IEEE) is currently the Principal Investigator of the NCSC-EPSRC Academic Centre of Excellence in Cyber Security Research, The University of Warwick; and a Professor of cyber systems engineering at WMG. He is a fellow of the Alan Turing Institute and the National Institute for Data Science and AI, U.K., where he is a Principal Investigator on a $5 million project developing trustworthy national identity to enable financial inclusion. He is a Co-Investigator of PETRAS, the National Centre of Excellence for IoT Systems Cyber Security and works with numerous banking organizations advising on security, privacy, and use of artificial intelligence. He has an international research reputation and extensive experience of institutional strategy development and interacting with external agencies. He has published over 250 peer-reviewed papers. He is a coauthor of the U.K. Security Breach Investigations Report 2010, supported by the Serious Organised Crime Agency and the Police Central e-crime Unit.



H. VINCENT POOR (Life Fellow, IEEE) received the Ph.D. degree in EECS from Princeton University in 1977. From 1977 to 1990, he was on the faculty of the University of Illinois at Urbana-Champaign. Since 1990, he has been on the faculty at Princeton, where he is currently the Michael Henry Strater University Professor. From 2006 to 2016, he served as the Dean of Princeton's School of Engineering and Applied Science. He has also held visiting appointments at several other universities, including most recently at Berkeley and Cambridge. His research interests are in the areas of information theory, machine learning and network science, and their applications in wireless networks, energy systems, and related fields. Among his publications in these areas is the forthcoming book *Advanced Data Analytics for Power Systems* (Cambridge University Press). Dr. Poor is a member of the National Academy of Engineering and the National Academy of Sciences, and he is a foreign member of the Chinese Academy of Sciences, the Royal Society, and other national and international academies. He has received the IEEE Alexander Graham Bell Medal in 2017.