Load Embeddings for Scalable AC-OPF Learning

Terrence W.K. Mak, *Member, IEEE,* Ferdinando Fioretto, and Pascal Van Hentenryck *Member, IEEE*

***Abstract*—AC Optimal Power Flow (AC-OPF) is a fundamental building block in power system optimization. It is often solved repeatedly, especially in regions with large penetration of renewable generation, to avoid violating operational limits. Recent work has shown that deep learning can be effective in providing highly accurate approximations of AC-OPF. However, deep learning approaches may suffer from scalability issues, especially when applied to large realistic grids. This paper addresses these scalability limitations and proposes a load embedding scheme using a 3-step approach. The first step formulates the load embedding problem as a bilevel optimization model that can be solved using a penalty method. The second step learns the encoding optimization to quickly produce load embeddings for new OPF instances. The third step is a deep learning model that uses load embeddings to produce accurate AC-OPF approximations. The approach is evaluated experimentally on large-scale test cases from the NESTA library. The results demonstrate that the proposed approach produces an order of magnitude improvements in training convergence and prediction accuracy.**

***Index Terms*—Deep learning, dimension reduction, bilevel optimization**

I. Introduction

The *AC Optimal Power Flow* (AC-OPF) is an optimization model that finds the most economical generation dispatch meeting the consumer demand, while satisfying the physical and operational constraints of the underlying power network [1]. The AC-OPF, together with its approximations and relaxations, constitutes a fundamental building block for power-system applications, including security-constrained OPF (e.g., [2]), transmission switching (e.g., [3]), capacitor placement (e.g., [4]), expansion planning (e.g., [5]), and stability-constrained OPF (e.g., [6]).

The non-convexity of the OPF limits the frequency of many operational tools. However, in practice, generation schedules are often required to be updated every few minutes [7]. Additionally, the integration of renewable energy and demand response programs [8], creates significant stochasticity in load and generation. Therefore, setting generation schedules based on historical data may lead to sub-optimal dispatches and, in the worst cases, causes voltage and/or stability issues. Balancing generation and load rapidly without sacrificing economical efficiency is thus an important challenge.

To cope with the OPF computational complexity, system operators typically solve OPF approximations (e.g., DC-OPF models) with a fast load flow solver (e.g., fast-decouple methods) to check reactive capabilities and voltage issues. While more efficient than solving AC-OPF problems, this process may lead to sub-optimal solutions, substantial economical losses, and possibly convergence issues.

Recently, an interesting line of research has focused on how to approximate AC-OPF using Deep Learning Networks (DNN) [9]–[11]. Once a neural network is trained, predictions can be computed in the order of milliseconds with a single forward pass through the network. Approximating OPF using deep neural networks can be seen as an empirical risk minimization problem under physical and engineering constraints [12], [13]. Preliminary results have been encouraging, showing that DNNs can approximate the generator set-points of AC-OPF with high accuracy. However, these learning models tend to have very large number of parameters, and these numbers scale with the size of the test case. This raises significant scalability and convergence issues during training, restricting the potential applicability of deep learning for AC-OPF.

This paper proposes a novel approach to address the scalability issues of deep learning approaches to AC-OPF. The proposed approach is a load embedding scheme that reduces the input dimension (i.e., the number of loads) of the deep learning network. The approach is based on the recognition that, in many circumstances, aggregating loads at adjacent buses does not fundamentally change the nature of AC-OPF. The load embedding scheme has two key components:

1) an optimization model for load aggregation that reduces the number of loads in an OPF instance, while staying close to the optimal AC-OPF cost;
2) a learning model for load embedding that, given loads for an AC-OPF instance, returns a set of encoded loads of smaller dimensions.

The load aggregation optimization is modeled as a bilevel optimization which is then approximated by replacing the lower level by a set of active, thermal, and voltage constraints. The resulting single-level model is then solved using a penalty method. The learning model for load embedding learns to mimic this optimization model and the paper explores both a baseline linear model and a DNN for learning to encode.

Once the load encoder has been learned, the OPF learning task can be performed using an architecture similar to the one in Figure 1. The encoder first computes a load embedding, which are then used as inputs to a DNN architecture in order to predict the active and reactive power of generators.[1] The encoder and AC-OPF learning models do not share parameters and are trained in sequence. The proposed approach first learns the encoder and then trains the AC-OPF DNN using the outputs of the learned encoder.

The approach has been evaluated on a wide range of NESTA test cases [14] and the results show that the proposed encoding can produce significant dimensionality reduction. The

T.W.K. Mak and P. Van Hentenryck are affiliated with the School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332. F. Fioretto is affiliated with the Electrical Engineering and Computer Science Department, Syracuse University, Syracuse, NY 13244. e-mail contacts: wmak@gatech.edu, ffiorett@syr.edu, pvh@isye.gatech.edu.

[1]The paper will show how to generalize it to voltage magnitudes as well.

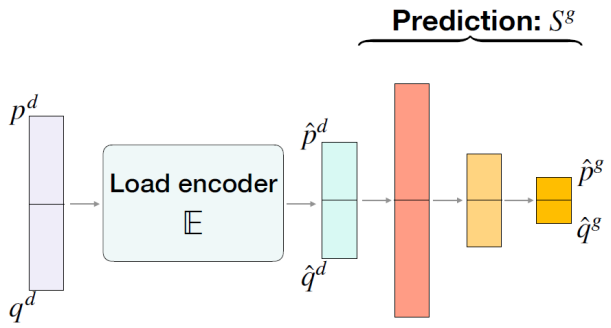

Fig. 1. The OPF-DNN Training Architecture with Encoder $\mathbb{E}$.

Model 1 $\mathcal{O}(\boldsymbol{S^d})$: AC Optimal Power Flow

input: $S_i^d \ \ \forall i \in N$

variables: $S_i^g, V_i \ \ \forall i \in N, \ \ S_{ij} \ \ \forall (i,j) \in E \cup E^R$

minimize: $$\sum_{i \in N} c_{2i}(\Re(S_i^g))^2 + c_{1i}\Re(S_i^g) + c_{0i} \quad (1)$$

subject to:
$$\angle V_s = 0, \ \ s \in N \quad (2)$$
$$v_i^l \leqslant |V_i| \leqslant v_i^u \ \ \forall i \in N \quad (3)$$
$$\theta_{ij}^l \leqslant \angle(V_i V_j^*) \leqslant \theta_{ij}^u \ \ \forall (i,j) \in E \quad (4)$$
$$S_i^{gl} \leqslant S_i^g \leqslant S_i^{gu} \ \ \forall i \in N \quad (5)$$
$$|S_{ij}| \leqslant s_{ij}^u \ \ \forall (i,j) \in E \cup E^R \quad (6)$$
$$S_i^g - S_i^d = \textstyle\sum_{(i,j) \in E \cup E^R} S_{ij} \ \ \forall i \in N \quad (7)$$
$$S_{ij} = Y_{ij}^* |V_i|^2 - Y_{ij}^* V_i V_j^* \ \ \forall (i,j) \in E \cup E^R \quad (8)$$

resulting architecture also exhibits significant improvements in convergence and prediction accuracy: indeed, the resulting AC-OPF learning can be an order of magnitude faster while, at the same time, improving the overall learning accuracy. The approach thus has the potential to provide an interesting and novel avenue to improve grid operations under significant penetration of renewable energy. The main contributions of this paper can be summarized as follows:

1) it proposes a bilevel optimization for load embedding;
2) it shows how to approximate the bilevel optimization by using a penalty method;
3) it presents learning models for load embeddings;
4) it proposes a scalable DNN architecture leveraging load embeddings for learning AC-OPF;
5) It demonstrates the potential of the proposed architecture on realistic test cases with thousands of buses and lines, showing improvements of an order of magnitude in training convergence and prediction accuracy.

The rest of the paper is organized as follows. Sections II and III present the related work and technical background. Section IV introduces the optimization models for load embedding. Section V presents that machine-learning models for load encoding. Section VI reviews the proposed scalable learning architecture for AC-OPF. Section VII reports the experimental results and Section VIII concludes the paper.

II. Related Work

Power network reduction techniques, such as Kron and Ward reduction [15] and Principle Component Anaylsis [16], have been widely used in the industry for more than 70 years. Early techniques focused on crafting simpler equivalent circuits to be used by system operators for analysis. With the advancement of computer technology and lower computation costs, complex reduction models became more feasible, e.g., models preserving line limits [17] and models handling dynamics [18]–[20]. While it is possible to use classical reduction techniques to reduce the size of power systems before applying a machine learning model, the learned model can only predict the reduced networks and with a potentially lower fidelity, compared to a learning approach on the original networks. *The approach proposed in this paper focuses on dimensionality reduction for deep neural networks*: its goal is to reduce the input size, and consequentially the number of learnable parameters, while retaining the prediction accuracy for the optimal generation dispatches and their costs.

Dimension reduction is an important and widely studied topic in machine learning, and reduction techniques have been successfully applied on various learning applications in power systems. For example, auto-encoders have been applied to predict renewable productions, e.g., wind [21] and solar [22] generations, and to detect false data injection attacks [23]. The proposed approach differs from general auto-encoder techniques on several aspects. First, the load embeddings are explicitly computed through a bilevel optimization model and not implicitly trained by an auto-encoder. Second, the computed load embeddings are AC-feasible, and their optimal power flows have the same cost as the original ones. Third, the reduced dimensionality is determined by optimization models instead of being chosen a-priori before training.

III. Background

This paper uses the rectangular form for complex power $S = p + jq$ and line/transformer admittance $Y = g + jb$, where p and q denote active and reactive powers, and g and b denote conductance and susceptance. Complex voltages are in polar form $V = ve^{j\theta}$, with magnitude $v = |V|$ and phase angle $\theta = \angle V$. Notation x^* is used to represent the complex conjugate of quantity x and notation $\hat{x}$ the prediction of quantity x. The reduction percentage from an original value v to a reduced value v^r is computed by formula: $100\% \times (v - v^r)/v$.

A. AC Optimal Power Flow

The AC Optimal Power Flow (OPF) determines the most economical generation dispatch balancing the load and generation in a power network (grid). A power network $\mathcal{N}$ is represented as a graph (N, E), where the set of nodes N represent buses and the set of edges E represent branches (including AC/DC transmission lines and transformers). Since edges in E are directed, E^R is used to denote arcs in the reverse direction. The AC power flow equations are expressed in terms of complex quantities for voltage V, admittance Y, and power S. Model 1 presents the AC OPF formulation, with variables and parameters in the complex domain. Superscripts

u and l are used to indicate upper and lower bounds for variables. The objective function $\mathcal{O}(\boldsymbol{S^g})$ captures the cost of the generator dispatch, with $\boldsymbol{S^g}$ denoting the vector of generator dispatch values $(S_i^g \mid i \in N)$. Constraint (2) sets the voltage angle of an arbitrary slack bus $s \in N$ to zero to eliminate numerical symmetries. Constraints (3) bound the voltage magnitudes, and constraints (4) limit the voltage angle differences for every branch. Constraints (5) enforce the generator output S_i^g to stay within its limits $[S_i^{gl}, S_i^{gu}]$. Constraints (6) impose the line flow limits s_{ij}^u on all the line flow variables S_{ij}. Constraints (7) capture Kirchhoff's Current Law enforcing the flow balance of generations S_i^g, loads S_i^d, and branch flows S_{ij} across every node. Finally, constraints (8) capture Ohm's Law describing the AC power flow S_{ij} across lines/transformers.

B. Deep Learning Models

Deep Neural Networks (DNNs) are learning architectures composed of a sequence of layers, each typically taking as inputs the results of the previous layer [24]. Feed-forward neural networks are basic DNNs where the layers are fully connected and the function connecting the layer is given by

$$\boldsymbol{y} = \pi(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}),$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is an input vector with dimension n, $\boldsymbol{y} \in \mathbb{R}^m$ is the output vector with dimension m, $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ is a matrix of weights, and $\boldsymbol{b} \in \mathbb{R}^m$ is a bias vector. Together, $\boldsymbol{W}$ and $\boldsymbol{b}$ define the trainable parameters of the network. The activation function π is non-linear (e.g., a rectified linear unit (ReLU)).

This paper uses the following OPF-DNN models from [9] to validate the quality of the proposed embedding scheme:

$$\begin{aligned} \boldsymbol{h}_1 &= \pi(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1) \\ \boldsymbol{h}_2 &= \pi(\boldsymbol{W}_2\boldsymbol{h}_1 + \boldsymbol{b}_2) \\ \boldsymbol{y} &= \pi(\boldsymbol{W}_3\boldsymbol{h}_2 + \boldsymbol{b}_3) \end{aligned} \tag{9}$$

where the input vector $\boldsymbol{x} = (\boldsymbol{p^d}, \boldsymbol{q^d})$ represents the vector of active and reactive loads, and the output vector $\boldsymbol{y}$ represents either the vector of active and reactive generation dispatch predictions $\boldsymbol{y} = (\boldsymbol{p^g}, \boldsymbol{q^g})$ or the vectors of voltage predictions $\boldsymbol{y} = (\boldsymbol{v}, \boldsymbol{\theta})$. Learning these DNN models consists in finding matrices $\boldsymbol{W_1}, \boldsymbol{W_2}, \boldsymbol{W_3}$, and the associated bias vectors $\boldsymbol{b_1}, \boldsymbol{b_2}, \boldsymbol{b_3}$, to make the output prediction $\hat{\boldsymbol{y}}$ close to the ground truth $\boldsymbol{y}$, as measured by a loss function $\mathbb{L}$, i.e., the mean squared error in this paper.

IV. Dimensionality Reduction by Load Embedding

This section motivates the concept of load embedding, which is formalized using a bilevel optimization model. For computational reasons, this bilevel model is relaxed into a single level model that is solved using a penalty method.

A. Motivation

Transmission systems are typically large and involve tens of thousands of buses and loads. The associated neural networks in (9) become large and difficult to train efficiently, since the sizes of their hidden layers are proportional to the input sizes. For example, in the model from Fioretto et al. [9], the dimensions of $\boldsymbol{h_1}$ can be as large $4|N|$ if there is a load at each bus. The proposed dimensionality reduction is motivated by the observation that, unless there is significant congestion or line power losses, moving a unit of load between two adjacent buses, will not have a major effect on the final dispatch. Therefore, training on a smaller version of the network that aggregates similar loads may be beneficial to the learning approach, while preserving the fidelity of the underlying operations. This section explores an encoder to perform such an aggregation.

B. The Bilevel Load-Embedding Model $\mathcal{M}_{BL}$

Let $\mathcal{O}^o$ be the optimal cost of the original OPF, $S_i^{g,o}$ the original dispatch of generator i, and $S_i^{d,o} = p_i^{d,o} + jq_i^{d,o}$ the original complex load i. The load-embedding model can be formulated as a bilevel optimization model $\mathcal{M}_{\text{BL}}$:

$$\min \quad \sum_{i \in N} \mathbb{1}(S_i^d \neq 0) \tag{10}$$

$$\text{s.t.} \quad (2) - (8) \qquad \text{(AC Power Flow)} \tag{11}$$

$$S_i^g = S_i^{g,o} \ \ \forall i \in N, \qquad \text{(Generation Equiv.)} \tag{12}$$

$$\sum_{i \in N} p_i^d = \sum_{i \in N} p_i^{d,o} \qquad \text{(Active Load Equiv.)} \tag{13}$$

$$\sum_{i \in N} q_i^d = \sum_{i \in N} q_i^{d,o} \qquad \text{(Reactive Load Equiv.)} \tag{14}$$

$$|\mathcal{O}(\boldsymbol{S^d}) - \mathcal{O}^o| \leqslant \beta \qquad \text{(Cost Equiv.)} \tag{15}$$

Its goal is to find the embedded loads (p_i^d, q_i^d) $(i \in N)$ which are the key decision variables. Objective (10) minimizes the number of nonzero loads using an indicator function. Constraints (11) imposes the power flow equations. Constraints (12) ensure that the generation dispatch remains the same, given that they are the targets of the learning task. Constraints (13) and (14) require the sum of the active and reactive loads to remain the same after the encoding. Together, these constraints ensure that the loads are AC-feasible for the original generation dispatch. However, they do not guarantee that they could not be served by a significantly better generator dispatch. This is the role of constraint (15) that ensures the encoded load vector $\boldsymbol{S^d} = \boldsymbol{p^d} + j\boldsymbol{q^d}$ induces an optimal flow with cost close to the original cost $\mathcal{O}^o$ (within a tolerance parameter β). This constraint uses an AC-OPF optimization as a subproblem, creating a bilevel model.

C. Load Embedding with Congestion Constraints: Model $\mathcal{M}_R$

Optimization model $\mathcal{M}_{\text{BL}}$ is challenging for two reasons: (1) it implicitly features discrete variables through the indicator variables in its objective; (2) it is a bilevel optimization problem. The first challenge can be addressed by replacing its discrete objective by a continuous expression that maximizes the square of the apparent power of each load, i.e.,

$$\max \quad \sum_{i \in N} [(p_i^d)^2 + (q_i^d)^2]. \tag{16}$$

Objective (16) encourages active and reactive loads to be aggregated without the need of binary variables.

Algorithm 1: Load Encoding

Input : $\mathcal{N}$: power grid data; ρ_v, ρ_s : penalty steps;
(β_v, β_s) : constraint tolerances;
β : cost tolerance;
i^u : max iteration limit.
Output: $\boldsymbol{S^d} = (\boldsymbol{p^d}, \boldsymbol{q^d})$

for $i = 0, 1, 2, \ldots, i^u$ **do**
  $\boldsymbol{S^d} \leftarrow (\boldsymbol{p^d}, \boldsymbol{q^d}) \leftarrow \mathcal{M}_{\mathrm{R}}(\beta_v, \beta_s)$
  if $|\mathcal{O}(\boldsymbol{S^d}) - \mathcal{O}^o| \leqslant \beta$ **then**
    break
  $\beta_v \leftarrow \rho_v \beta_v, \quad \beta_s \leftarrow \rho_s \beta_s$

The second challenge can be addressed by replacing constraint (15) by proxy constraints that characterize the OPF. Indeed, in the original OPF, a number of voltage and thermal constraints are binding. Imposing constraints on the associated voltages and flows will help in keeping the optimal cost close to the original cost. Let N^l and N^u be the set of buses with binding lower and upper constraints on voltages, and E^u be the set of lines with binding thermal limit constraints. Constraint (15) can be relaxed and reformulated as:

$$|v_i - v_i^u| \leqslant \beta_v, \forall i \in N^u \qquad \text{(Volt. Congestion)} \quad (17)$$
$$|v_i - v_i^l| \leqslant \beta_v, \forall i \in N^l \qquad \text{(Volt. Congestion)} \quad (18)$$
$$||S_{ij}| - s_{ij}^u| \leqslant \beta_s, \forall (i,j) \in E^u \qquad \text{(Line Congestion)} \quad (19)$$

where β_v and β_s are the tolerance parameters for the tightness of the original binding constraints. The relaxed model $\mathcal{M}_{\mathrm{R}}$ is then defined as:

$$\begin{aligned} \max \quad & \sum_{i \in N} [(p_i^d)^2 + (q_i^d)^2] & \\ \text{s.t.} \quad & (2) - (8) & \text{(AC Power Flow)} \\ & (12) - (14) & \text{(Equiv. Constr.)} \\ & (17) - (19) & \text{(Congestion Constr.)} \end{aligned}$$

D. Load Embedding with a Penalty Method: Model $\mathcal{M}_P$

Model $\mathcal{M}_{\mathrm{R}}$ requires the choice of tolerance parameters β_v and β_s. If these tolerances are too tight, it may not be possible to aggregate loads effectively. If they are too loose, the resulting predictions may be inaccurate. To overcome this difficulty, this paper uses a penalty method.[2] The resulting model $\mathcal{M}_{\mathrm{P}}(\beta_v, \beta_s)$ becomes

$$\begin{aligned} \max \quad & \sum_{i \in N} [(p_i^d)^2 + (q_i^d)^2] + \beta_v \sum_{i \in N^u} \|v_i - v_i^u\|^2 + \\ & \beta_v \sum_{i \in N^l} \|v_i - v_i^l\|^2 + \beta_s \sum_{(i,j) \in E^u} \| |S_{ij}| - s_{ij}^u \|^2 \\ \text{s.t.} \quad & (2) - (8) \text{ and } (12) - (14) \end{aligned}$$

and it can be solved iteratively by increasing β_v and β_s until (15) is satisfied, using Algorithm 1.

[2]Alternatively, it is possible to use an Augmented Lagrangian Method. Experimental results have shown that the encoding quality is similar but solving times were slightly longer for the latter.

V. Learning to Encode

Algorithm 1 computes the best load embedding for a load profile $\boldsymbol{S^d}$. It is appropriate to provide embeddings as inputs when training and validating machine-learning models. But, obviously, Algorithm 1 cannot be used at prediction time, since this would defeat the purpose of using machine learning to speed up OPF computations. To overcome this difficulty, the paper proposes to *learn the encoder*, and explores two learning schemes: (1) a linear regression

$$\mathbb{E}_L(\boldsymbol{x}) = \boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b},$$

and (2) a deep neural network similar to (9)

$$\mathbb{E}_F(\boldsymbol{x}) = \boldsymbol{z} = \pi\{\boldsymbol{W}_3\pi[\boldsymbol{W}_2\pi(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2] + \boldsymbol{b}_3\}.$$

The input vector is the load vector $\boldsymbol{x} = (\boldsymbol{p^d}, \boldsymbol{q^d})$ and the output vector is the embedded load vector $\boldsymbol{z} = (\hat{\boldsymbol{p}}^{\boldsymbol{d}}, \hat{\boldsymbol{q}}^{\boldsymbol{d}})$. The structure of the output, i.e., the embedded load vector, is obtained by removing the loads that are *relocated in all the training instances*. For the full NN-encoder, the dimension of the first and second layers are set to twice of the dimensions of the input and the output vectors respectively. For a data set collection $\mathbb{D} = \{(\boldsymbol{x}^i, \boldsymbol{z}^i) : i \in [1 \ldots n]\}$ with n test cases where the outputs $\boldsymbol{z}^i$ are computed using Algorithm 1, the goal of the learning task is to find the model parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ that minimize the empirical risk function:

$$\min_{\boldsymbol{W}, \boldsymbol{b}} \sum_{(\boldsymbol{x}^i, \boldsymbol{z}^i) \in \mathbb{D}} \mathbb{L}(\mathbb{E}_m(\boldsymbol{x}^i), \boldsymbol{z}^i). \quad (20)$$

where $m \in \{L, F\}$ is used to discriminate the model adopted. Once the training is complete, any output element $\hat{\boldsymbol{p}}^{\boldsymbol{d}}$ or $\hat{\boldsymbol{q}}^{\boldsymbol{d}}$ that is always zero (i.e., the load is always aggregated elsewhere) can be removed in order to provide a more compact input to OPF predictor.

VI. Scalable OPF Learning

Once the load encoder has been learned, the OPF learning task is performed using the architecture in Figure 1. NN layers (tensors) are represented by rectangular boxes and arrows represent connections between layers. The architecture uses fully connected layers with ReLU as the activation functions. Notice that the encoder is pre-trained, so the learning task will not affect its parameters. The dimensions of h_1 and h_2 of the OPF layers are set to twice of the dimension of the input and the output vectors respectively as in [9].

a) Leveraging Physical Constraints: Unless required to, the encoder does not preserve the values of many physical parameters, including phase angles, voltage magnitudes, and line flows. However, these physical values on the reduced network, which are available as a result of Algorithm 1, are important to improve prediction accuracy using, for instance, a Lagrangian dual approach as in [9]. Although they are not useful for prediction purposes, they can enhance the loss function, which can now include penalities for the violations of physical constraints.

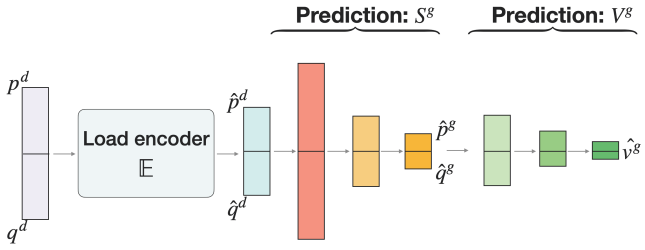

Fig. 2. The extended OPF-DNN Training Architecture with Encoder $\mathbb{E}$.

b) Predicting the Generator Setpoints: The approach can be extended to predict the voltage magnitude setpoints $\boldsymbol{v}^g$ for each of the generators. Figure 2 shows how to add a new DNN to the existing OPF-DNN architecture to predict the voltage magnitudes. The added DNN takes, as input, the existing output layer $\boldsymbol{y} = (\hat{\boldsymbol{p}}^g, \hat{\boldsymbol{q}}^g)$, is defined as

$$\mathbb{V}^g(\boldsymbol{y}) = \hat{\boldsymbol{v}}^g = \pi\{\boldsymbol{W}_3'\pi[\boldsymbol{W}_2'\pi(\boldsymbol{W}_1'\boldsymbol{y} + \boldsymbol{b}_1') + \boldsymbol{b}_2'] + \boldsymbol{b}_3'\},$$

and produces the predictions $\hat{\boldsymbol{v}}^g$ for the generator voltage magnitudes. The learning task has the additional parameters $\boldsymbol{W}'$ and $\boldsymbol{b}'$, and the additional loss function

$$\min_{\boldsymbol{W}',\boldsymbol{b}'} \sum_{(\boldsymbol{y}^i,\boldsymbol{v}^{gi})\in\mathbb{D}} \mathbb{L}(\mathbb{V}^g(\boldsymbol{y}^i), \boldsymbol{v}^{gi}). \tag{21}$$

VII. Experimental Evaluation

This section presents the experimental results.

a) Experimental Setting: The experiments were performed on various NESTA [14] benchmarks, and Algorithm 1 was implemented on top of PowerModels.jl [25], a state-of-the-art Julia package for solving or approximating AC-OPF. The tolerance β was set to 0.5%, $i^u = 500$, and parameters ρ_v and ρ_s were both set to 1.5.

The OPF data sets were generated by varying the load profiles of each benchmark network from 80% to 120% of their original (complex) load values, with a step size of 0.02%, giving a maximum of 2000 test cases for every benchmark network. For each test case, to create enough diversity, every load is perturbed with random noise from the polar Laplace distribution whose parameter λ is set to 10% of the apparent power. Higher values of noise typically create infeasible test cases. Test cases, with no feasible AC solutions, were removed from the data set. The remaining cases were split with 80%-20% ratio for training and validation.

The OPF-DNN models, as well as the encoding models ($\mathbb{E}_L/\mathbb{E}_F$), were implemented using PyTorch [26] and run with Python 3.6. They used the Mean Squared Error (MSE) as loss function. The training was performed using Tesla-V100 GPUs with 16GBs HBM2 ram on machines with Intel CPU cores at 2.1GHz. The training used Averaged Stochastic Gradient Descent (ASGD) with learning rate 0.001 and 3000 epochs.

b) Compression Ratios and Efficiency of Algorithm 1: This section studies the compression ratios, the accuracy loss, and the efficiency of Algorithm 1. Table I shows the active and reactive load compression ratios (in percentages), the OPF error (in percentage), and the CPU time of Algorithm 1. Let c_p and c_q be the number of (non-zero) active and reactive loads, c_p^e and c_q^e be the number of (non-zero) embedded active and reactive loads by running Algorithm 1, and $\mathcal{O}^o$ and $\mathcal{O}^e$ be the original and "embedded" OPF costs. The compression ratios for active and reactive loads, and the OPF error, are given by $100\% \times (c_p - c_p^e)/c_p$, $100\% \times (c_q - c_q^e)/c_q$, and $100\% \times (\mathcal{O}^o - \mathcal{O}^e)/\mathcal{O}^o$ respectively. Almost all NESTA benchmarks achieve a load compression ratio (both active & reactive) over 50%, and the OPF errors are well within the prescribed 0.5% of the original cost $\mathcal{O}^o$. Encoding large benchmarks with Algorithm 1 may take time as the results indicate, but this is performed off-line before training.

Table II shows, for each test case, the dimension (i.e., $c_p + c_q$) for the load vector $(\boldsymbol{p^d}, \boldsymbol{q^d})$ in the original data

TABLE I
Evaluation of Load Embeddings.

Network Benchmarks (Bus # / ID)	Load Compression Active (%)	Reactive (%)	Joint (%)	OPF Error (%)	CPU Time (sec.)
14_ieee	63.64	81.82	72.73	0.23	3.66
24_ieee_rts	70.59	52.94	61.76	0.12	1.54
29_edin	68.97	37.93	53.45	0.06	4.38
30_as	90.48	80.95	85.71	0.26	1.34
30_fsr	70.00	75.00	72.50	0.27	1.20
30_ieee	85.71	90.48	88.10	0.29	7.88
39_epri	61.90	66.67	64.29	0.04	1.22
57_ieee	92.86	78.57	85.71	0.02	2.01
73_ieee_rts	76.47	70.59	73.53	0.30	3.37
89_pegase	28.57	40.00	34.29	0.04	3.39
118_ieee	75.76	87.78	81.48	0.27	109.59
162_ieee_dtc	67.26	84.52	74.62	0.16	265.93
189_edin	80.39	86.27	83.33	0.48	117.93
240_wecc	63.31	56.72	60.07	0.46	1277.18
1354_pegase	77.27	72.38	74.85	0.27	1865.80
1394sop_eir	95.04	74.23	84.67	0.23	13746.75
1397sp_eir	93.94	79.55	86.74	0.44	1165.71
1460wp_eir	88.43	71.27	79.85	0.39	21263.20
1888_rte	51.86	50.40	51.13	0.07	703.14
1951_rte	74.65	64.41	69.55	0.06	523.30
2848_rte	43.14	42.37	42.76	0.25	661.36
2868_rte	62.99	50.95	57.00	0.11	1833.68
3012wp_mp	90.53	88.06	89.32	0.35	1204.94
3375wp_mp	88.12	87.03	87.59	0.23	10133.59

TABLE II
OPF-DNN: Original & Reduced Input dimension

Network	Original dim.	Reduced dim.	Reduction %
14_ieee	22	11	50%
30_ieee	42	13	69%
39_epri	42	29	31%
57_ieee	84	26	69%
73_ieee_rts	102	88	14%
89_pegase	70	55	21%
118_ieee	198	198	0%
162_ieee_dtc	226	143	37%
189_edin	1244	336	73%
1394_sop_eir	524	207	60%
1460_wp_eir	536	536	0%
1888_rte	2000	1555	22%
2848_rte	3022	2310	24%
2868_rte	3102	2221	28%
3012wp_mp	4542	2043	55%
3375wp_mp	4868	2109	57%

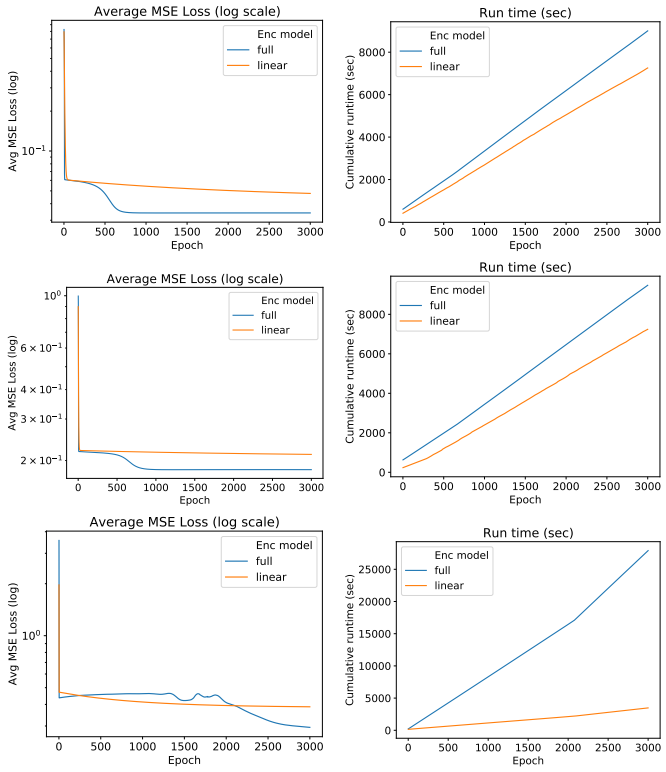

Fig. 3. Average MSE error (log scale) and Cumulative Runtime (sec) during the encoder training phase. Top: 1394_sop_eir, middle: 1460wp_eir, bottom: 3375wp_mp.

set and compares it to the reduced dimension for the load vector ($\hat{\boldsymbol{p}}^{\boldsymbol{d}}$,$\hat{\boldsymbol{q}}^{\boldsymbol{d}}$). This reduced dimension is computed by running algorithm Algorithm 1 on the ($\approx$ 2000) test cases for each benchmark data set and removing the loads that are always assigned to zero. Remarkably, given the wide range of considered loads, many of the test cases (except the 118 and 1460 cases) achieve a significant dimensionality reduction. Of particular interest are the large RTE test cases whose dimensions are reduced by 24% and 28% and the large `wp_mp` test cases whose dimensions are reduced by 55% and 57%.

c) Encoder Learning: This section presents results on the learning of the encoder. Figure 3 shows, on three large test cases, the Mean Squared Errors (averaged by the number of training cases) and cumulative runtimes in seconds for training the linear encoder $\mathbb{E}_L$ and the full-NN encoder $\mathbb{E}_F$. The linear encoder trains faster than the full-NN encoder, but its prediction errors, albeit small, are almost always larger than the full-NN encoder.

d) OPF Prediction Errors: This section shows that learning with encoders almost always reduces prediction errors, which is interesting in its own right. Table III and Table IV depict the prediction results for three variants of OPF-DNN models on the testing data set: a) no encoder, b) the OPF-DNN architecture with encoder $\mathbb{E}_L$, and c) the OPF-DNN architecture with encoder $\mathbb{E}_F$. Table III reports the averaged L1-losses $\|\cdot\|_1$ for the *main predictions*, i.e., the active and

TABLE III
PREDICTION ERRORS FOR GENERATOR DISPATCH: L1 ERROR (P.U.).

Network	No Enc.	Linear Enc.	Full Enc.
14_ieee	0.0065	0.0057	**0.0050**
30_ieee	0.0041	**0.0033**	0.0038
39_epri	0.2536	0.0632	**0.0422**
57_ieee	0.0433	0.0522	**0.0130**
73_ieee_rts	0.0602	**0.0178**	0.0676
89_pegase	0.1807	**0.0243**	0.0360
118_ieee	0.0504	0.0108	**0.0063**
162_ieee_dtc	0.1622	0.0493	**0.0329**
189_edin	0.0209	0.0117	**0.0075**
1394_sop_eir	0.0041	0.0039	**0.0029**
1460_wp_eir	0.0129	0.0114	**0.0055**
1888_rte	0.1964	**0.0792**	0.2046
2848_rte	0.0376	0.0125	**0.0085**
2868_rte	0.025	**0.0095**	0.2026
3012_wp_mp	**0.0420**	0.0490	0.0486
3375wp_mp	0.0483	0.0252	**0.0212**

TABLE IV
PREDICTION ERRORS OF OPF-DNN: AVG. MEAN SQUARED ERROR (P.U.)

Network	No Enc.	Linear Enc.	Full Enc.
14_ieee	0.0003	0.0003	**0.0002**
30_ieee	0.0010	**0.0007**	**0.0007**
39_epri	0.1443	0.0092	**0.0046**
57_ieee	0.0061	0.0080	**0.0007**
73_ieee_rts	0.0211	**0.0079**	0.0271
89_pegase	0.3152	**0.0031**	0.0054
118_ieee	0.0264	0.0051	**0.0034**
162_ieee_dtc	0.1321	0.0069	**0.0036**
189_edin	0.0042	0.0019	**0.0010**
1394_sop_eir	0.0021	0.0007	**0.0004**
1460_wp_eir	0.0043	0.0034	**0.0015**
1888_rte	0.2890	**0.0888**	0.3194
2848_rte	0.0189	0.0042	**0.0031**
2868_rte	0.0071	**0.0023**	0.4699
3012_wp_mp	**0.0151**	0.0185	0.0184
3375wp_mp	0.0722	0.0139	**0.0074**

reactive generation dispatches, using the formula

$$\frac{1}{|T|}\sum_{t\in T}\left[\frac{\|\hat{\boldsymbol{p}}^{\boldsymbol{g}}_{\boldsymbol{t}},\boldsymbol{p}^{\boldsymbol{g}}_{t}\|_1/|N^G| + \|\hat{\boldsymbol{q}}^{\boldsymbol{g}}_{\boldsymbol{t}},\boldsymbol{q}^{\boldsymbol{g}}_{t}\|_1/|N^G|}{2}\right]$$

where N^G is the set of generators and T is the set of testing data. Table IV complements Table III by reporting the combined Mean Squared Error losses (MSE) on all prediction variables (including the *indirect* voltage support variables), using the formula

$$\frac{1}{|T|}\sum_{t\in T}\Big[\ \frac{\text{MSE}(\hat{\boldsymbol{p}}^{\boldsymbol{g}}_{\boldsymbol{t}},\boldsymbol{p}^{\boldsymbol{g}}_{t}) + \text{MSE}(\hat{\boldsymbol{q}}^{\boldsymbol{g}}_{t},\boldsymbol{q}^{\boldsymbol{g}}_{t})}{2} + \frac{\text{MSE}(\hat{\boldsymbol{v}}_t,\boldsymbol{v}_t) + \text{MSE}(\hat{\boldsymbol{\theta}}_t,\boldsymbol{\theta}_t)}{2}\ \Big]$$

The results demonstrate the effectiveness of the proposed encoders, which yield predictors with smaller errors. Interestingly, even for the 118 bus and 1460 bus benchmarks, which have no dimensionality reduction, the generation dispatch errors are reduced by an order of magnitude.

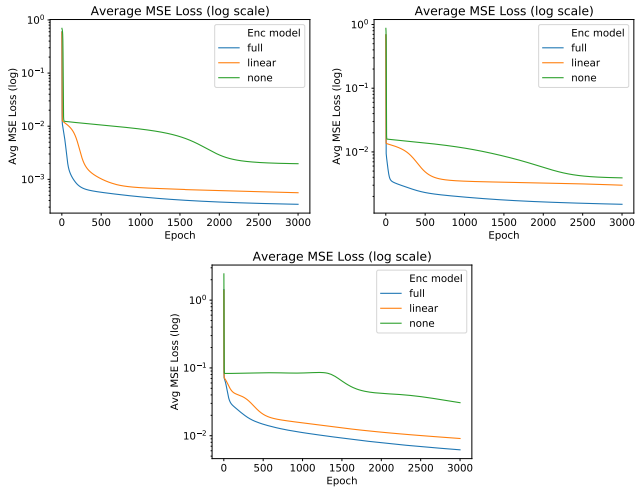

Fig. 4. Average MSE error (log scale) during training phase. Top-left: 1394_sop_eir, top-right: 1460wp_eir, bottom: 3375wp_mp.

e) Training Convergence and Speed: The key motivation of this paper is of course to speed up the learning task. This section demonstrates that the OPF-DNN architecture with load encoding quickly converges to an accuracy that is an order of magnitude better than full OPF-DNN architecture. Figure 4 shows the combined MSE losses (in log scale) for the three OPF-DNN architectures during the training phase for the 1394, 1460, and 3375 bus benchmarks. The results are averaged by the number of training cases as in previous sections. The full NN-encoder is almost an order of magnitude more accurate than the base model, and consistently better than the linear encoder model. Both load-encoding architectures outperforms the base model in the early convergence period (within 500 epochs), and a significant convergence gap still exists even after the error curves have flattened (e.g., after 2500 epochs). These results indicate that load reduction yields both a better training convergence and smaller prediction errors.

f) Predicting Generator Voltage Setpoints: Table V is the counterpart to Table III: it reports, on selected benchmarks, the averaged L1-losses $\|\cdot\|_1$ for the *generator voltage setpoint predictions* using the formula: $\frac{1}{|T|}\sum_{t\in T}\frac{\|\hat{\boldsymbol{v}}_t^g,\boldsymbol{v}_t^g\|_1}{|N^G|}$ where N^G is the set of generators and T is the set of testing instances. The results again demonstrate the effectiveness of the proposed encoders, which yield predictors with smaller errors when instances are large. Figure 5 shows the MSE losses (in log scale) for the generator voltage predictions during the training phase for the 39, 189, and 2848_rte bus benchmarks. The results are largely similar to prior results and show the significant benefits on load embeddings on the larger test cases.

g) OPF-DNN with Constraints: To analyze whether load encoding is useful across different learning models, this section reports accuracy and convergence results for the OPF-DNN with constraints and Lagrangian multipliers (proposed by Fioretto et al. [9]). Table VI and Table VII depict the validation results for three variants of OPF-DNN models: a) with no load encoder; b) with a trained linear encoder $\mathbb{E}_L$; and c) with a trained full NN-encoder $\mathbb{E}_F$. Figure 6 shows the combined MSE losses (in log scale), and "Mem Err" is used for test cases exceeding the GPU memory limits.

TABLE V
PREDICTION ERRORS FOR GENERATOR VOLTAGE MAG.: L1 ERROR (P.U.).

Network	No Enc.	Linear Enc.	Full Enc.
14_ieee	0.0032	0.0028	**0.0022**
30_ieee	**0.0026**	0.0040	0.0039
39_epri	0.0385	0.0079	**0.0066**
57_ieee	0.0045	0.0032	**0.0024**
73_ieee_rts	0.0259	**0.0054**	0.0349
89_pegase	0.0047	**0.0038**	0.0062
118_ieee	0.0272	0.0026	**0.0024**
162_ieee_dtc	0.0114	0.0044	**0.0031**
189_edin	0.0068	0.0039	**0.0038**
1394_sop_eir	0.0033	0.0041	**0.0029**
2848_rte	0.1534	**0.0017**	**0.0017**
3375wp_mp	0.0048	0.0037	**0.0035**

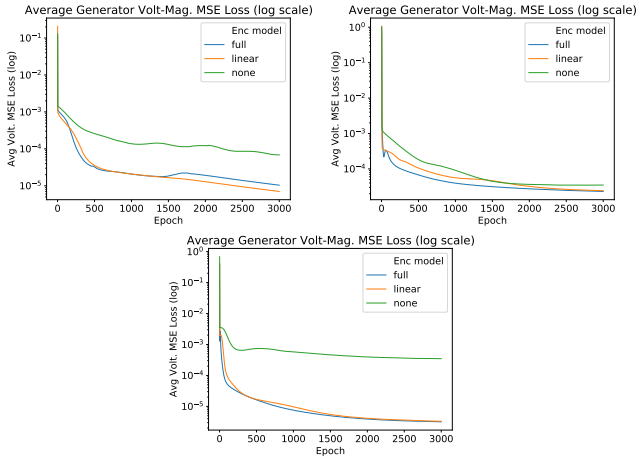

Fig. 5. Average MSE error (log scale) during training phase for generator voltage magnitude predictions. Top-left: 39_epri, Top-right: 189_edin, bottom: 2848_rte

TABLE VI
VALIDATION RESULTS FOR OPF-DNN WITH CONSTRAINTS: GENERATOR DISPATCH L1 ERROR (P.U.).

Network	No Enc.	Linear Enc.	Full Enc.
14_ieee	0.0062	0.0053	**0.0049**
30_ieee	0.0041	**0.0033**	0.0037
39_epri	0.2480	**0.0794**	0.7180
57_ieee	0.0461	0.0259	**0.0118**
73_ieee_rts	0.0600	**0.0167**	0.0252
89_pegase	0.1078	**0.0262**	0.0287
118_ieee	0.0646	0.0099	**0.0061**
162_ieee_dtc	0.3125	0.0493	**0.0411**
189_edin	0.0203	0.0116	**0.0074**
1394_sop_eir	0.0041	0.0039	**0.0028**
1460_wp_eir	0.0129	0.0113	**0.0055**
1888_rte	0.1963	**0.0783**	0.2040
2848_rte	0.0870	0.0123	**0.0084**
2868_rte	Mem Err	0.0096	Mem Err
3012_wp_mp	**0.0421**	0.0491	0.0482
3375wp_mp	Mem Err	0.0247	**0.0215**

The results are largely similar to those of the previous subsections. The learning models with encoders consistently deliver smaller prediction errors in the validation phase, and outperform the base model during the training phase.

TABLE VII
VALIDATION RESULTS FPR OPF-DNN WITH CONSTRAINTS: AVG. MEAN SQUARED ERROR (P.U.)

Network	No Enc.	Linear Enc.	Full Enc.
14_ieee	0.0003	0.0003	0.0003
30_ieee	0.0010	**0.0007**	**0.0007**
39_epri	0.1396	**0.0128**	0.9407
57_ieee	0.0065	0.0030	**0.0005**
73_ieee_rts	0.0110	**0.0077**	0.0210
89_pegase	0.1038	**0.0038**	0.0040
118_ieee	0.0378	0.0049	**0.0028**
162_ieee_dtc	0.4604	0.0091	**0.0060**
189_edin	0.0041	0.0020	**0.0011**
1394_sop_eir	0.0021	0.0006	**0.0004**
1460_wp_eir	0.0043	0.0033	**0.0015**
1888_rte	0.2886	**0.0874**	0.3186
2848_rte	0.0661	0.0041	**0.0031**
2868_rte	Mem Err	0.0023	Mem Err
3012_wp_mp	**0.0151**	0.0185	0.0182
3375wp_mp	Mem Err	0.0147	**0.0080**

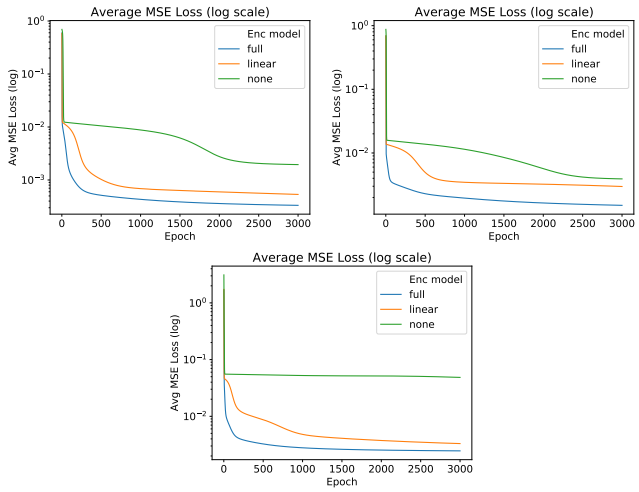

Fig. 6. Average MSE error (log scale) during training phase. Top-left: 1394_sop_eir, top-right: 1460_wp_eir, bottom: 2848_rte.

VIII. CONCLUSION

This paper studied how to improve the scalability of deep neural networks for learning the active and reactive powers of generatorsin AC-OPF. To address computational issues that arise in learning AC-OPF over large networks, this paper proposed a load encoding scheme for dimensionality reduction and its associated deep learning architecture. The load encoding scheme consists of (1) an optimization model to aggregate loads for each instance; and (2) a deep learning model that approximates the load encoding. The learned encoder can then be included in a deep learning architecture for AC-OPF and produces an order of magnitude improvement in training convergence and prediction accuracy of realistic test cases with thousands of buses and lines. These results show the potential of the approach in improving the scalability of deep learning for power systems.

REFERENCES

[1] B. H. Chowdhury and S. Rahman, "A review of recent advances in economic dispatch," *IEEE Transactions on Power Systems*, vol. 5, no. 4, pp. 1248–1259, Nov 1990.
[2] A. Monticelli, M. Pereira, and S. Granville, "Security-constrained optimal power flow with post-contingency corrective rescheduling," *IEEE Transactions on Power Systems*, vol. 2, no. 1, pp. 175–180, 1987.
[3] E. B. Fisher, R. P. O'Neill, and M. C. Ferris, "Optimal transmission switching," *IEEE Transactions on Power Systems*, vol. 23, no. 3, pp. 1346–1355, Aug 2008.
[4] M. E. Baran and F. F. Wu, "Optimal capacitor placement on radial distribution systems," *IEEE Transactions on Power Delivery*, vol. 4, no. 1, pp. 725–734, Jan 1989.
[5] Niharika, S. Verma, and V. Mukherjee, "Transmission expansion planning: A review," in *International Conference on Energy Efficient Technologies for Sustainability*, April 2016, pp. 350–355.
[6] D. Gan, R. J. Thomas, and R. D. Zimmerman, "Stability-constrained optimal power flow," *IEEE Transactions on Power Systems*, vol. 15, no. 2, pp. 535–540, 2000.
[7] J. Tong and H. Ni, "Look-ahead multi-time frame generator control and dispatch method in PJM real time operations," in *IEEE Power and Energy Society General Meeting*, July 2011.
[8] P. Scott and S. Thiébaux, "Distributed multi-period optimal power flow for demand response in microgrids," in *2015 ACM International Conference on Future Energy Systems*, 2015, pp. 17–26.
[9] F. Fioretto, T. W. Mak, and P. Van Hentenryck, "Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 630–637.
[10] Z. Yan and Y. Xu, "Real-time optimal power flow: A lagrangian based deep reinforcement learning approach," *IEEE Transactions on Power Systems*, vol. 35, no. 4, pp. 3270–3273, 2020.
[11] X. Pan, T. Zhao, and M. Chen, "Deepopf: Deep neural network for dc optimal power flow," in *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2019, pp. 1–6.
[12] Y. Ng, S. Misra, L. Roald, and S. Backhaus, "Statistical learning for DC optimal power flow," in *Power Systems Computation Conference*, 2018.
[13] D. Deka and S. Misra, "Learning for DC-OPF: Classifying active sets using neural nets," in *2019 IEEE Milan PowerTech*, June 2019.
[14] C. Coffrin, D. Gordon, and P. Scott, "NESTA, the NICTA energy system test case archive," *CoRR*, vol. abs/1411.0359, 2014. [Online]. Available: http://arxiv.org/abs/1411.0359
[15] J. B. Ward, "Equivalent circuits for power-flow studies," *Transactions of the American Institute of Electrical Engineers*, vol. 68, no. 1, pp. 373–382, July 1949.
[16] H. K. Amchin and E. T. B. Gross, "Analyses of subsequent faults," *Electrical Engineering*, vol. 71, no. 5, pp. 420–420, 1952.
[17] W. Jang, S. Mohapatra, T. J. Overbye, and H. Zhu, "Line limit preserving power system equivalent," in *2013 IEEE Power and Energy Conference at Illinois (PECI)*, 2013, pp. 206–212.
[18] S. Y. Caliskan and P. Tabuada, "Kron reduction of power networks with lossy and dynamic transmission lines," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 2012, pp. 5554–5559.
[19] I. P. Nikolakakos, H. H. Zeineldin, M. S. El-Moursi, and J. L. Kirtley, "Reduced-order model for inter-inverter oscillations in islanded droop-controlled microgrids," *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 4953–4963, 2018.
[20] Z. Jiang, N. Tong, Y. Liu, Y. Xue, and A. G. Tarditi, "Enhanced dynamic equivalent identification method of large-scale power systems using multiple events," in *Power Systems Computation Conference (PSCC)*, 2020.
[21] S. Tasnim, A. Rahman, A. M. T. Oo, and M. E. Haque, "Autoencoder for wind power prediction," *Renewables: Wind, Water, and Solar*, vol. 4, no. 1, p. 6, 2017.
[22] A. Gensler, J. Henze, B. Sick, and N. Raabe, "Deep learning for solar power forecasting — an approach using autoencoder and lstm neural networks," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2016, pp. 2858–2865.
[23] A. Kundu, A. Sahu, E. Serpedin, and K. Davis, "A3d: Attention-based auto-encoder anomaly detector for false data injection attacks," in *Power Systems Computation Conference (PSCC)*, 2020.
[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.
[25] C. Coffrin, R. Bent, K. Sundar, Y. Ng, and M. Lubin, "Powermodels.jl: An open-source framework for exploring power flow formulations," in *PSCC*, June 2018.
[26] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.