UNIVERSITY OF CALIFORNIA
SANTA CRUZ

POWER FLOW ANALYSIS AND OPTIMAL POWER FLOW WITH PHYSICS-INFORMED DEEP LEARNING

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL AND COMPUTER ENGINEERING

by

Kejun Chen

March 2025

The Dissertation of Kejun Chen
is approved:

Dr. Yu Zhang, Chair

Dr. Patrick E. Mantey

Dr. Leila Parsa

Peter Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures v

List of Tables vii

Abstract ix

Acknowledgments xii

1 Introduction 1
- 1.1 Overview . . . 1
- 1.2 Literature Review . . . 8
  - 1.2.1 PPF analysis . . . 8
  - 1.2.2 AC-OPF analysis . . . 12
- 1.3 Contributions . . . 19

2 Probabilistic Power Flow Analysis 21
- 2.1 Problem Formulation . . . 22
  - 2.1.1 System Description . . . 22
  - 2.1.2 AC-PF Problem Formulation . . . 23
  - 2.1.3 From PF Analysis to PPF Analysis . . . 24
- 2.2 Physics-guided Residual Learning for PPF Analysis . . . 25
  - 2.2.1 Designed Framework inspired by Residual Learning . . . 26
  - 2.2.2 Initialization Schemes for the Shortcut Layer . . . 28
  - 2.2.3 Simulation Results . . . 32
- 2.3 Variation-cognizant PPF via Multi-task Learning . . . 44
  - 2.3.1 Proposed methods for PF Analysis . . . 44
  - 2.3.2 Simulation Results . . . 46
- 2.4 Summary . . . 52

3 Optimal Power Flow Analysis 54
- 3.1 AC-OPF Problem Formulation . . . 55
- 3.2 Unsupervised Learning Framework for AC-OPF Analysis . . . 56

3.2.1 AC-OPF problem via Lagrangian Function 56
3.2.2 NN Learning framework for AC-OPF Analysis 58
3.2.3 Combine NN training with Lagrangian Duality 60
3.2.4 Simulation Results 61
3.3 Gradient Estimation method to Accelerate NN Training for AC-OPF . . 68
3.3.1 Semi-supervised Learning via Data Augmentation 69
3.3.2 Gradient Computation 72
3.3.3 Simulation Results 83
3.4 Summary 91

4 Conclusion and Future work **96**
4.1 Conclusion 96
4.2 Future Work 99
4.2.1 Literature Review 99
4.2.2 Problem Formulation 101

List of Figures

2.1 The residual block structure and the proposed NN structure based on residual learning. 27
2.2 The MAPEs of phase angle calculations on the IEEE-118 bus system. . 38
2.3 The MAPEs of voltage magnitude calculations on the IEEE-118 bus system. 38
2.4 Voltage angle of bus 1 for the IEEE-300 bus system. 40
2.5 Voltage magnitude of bus 16 for the IEEE-300 bus system. 40
2.6 The training loss evolution process (starting from when the first epoch's training is done) on the IEEE-30 bus system. 42
2.7 The training loss evolution process (starting from when the first epoch's training is done) on the IEEE-118 bus system. 42
2.8 Weights of the shortcut connection linear layer of the *Random* method for the IEEE-30 bus system. 43
2.9 Weights of the shortcut connection linear layer of the *Data-driven* method for the IEEE-30 bus system. 43
2.10 The Wasserstein distance of voltage magnitude distributions for the IEEE-300 bus system. 51
2.11 The Wasserstein distance of branch flows distributions for the IEEE-300 bus system. 51

3.1 The proposed unsupervised learning framework for solving AC-OPF. . . 59
3.2 The proposed unsupervised learning framework using VS_2 for solving AC-OPF. 62
3.3 The boxplot of the active power generations on the IEEE-30 bus system. 65
3.4 The training loss trajectory of different gradient estimation methods on the IEEE-118 bus system. The learning rate is set to 0.0005 for the initial 90 epochs, then reduced to 0.0001 for the last 30 epochs. 92
3.5 The total training loss trajectory of the proposed method M_4 over epochs on the IEEE-118 bus system. The learning rate is 0.0005 in the first 90 epochs and 0.0001 in the last 30 epochs. 92

3.6 The total training loss evolution process of the proposed method M_4 over epochs on the PEGASE-1354 bus system. The learning rate is 0.0005 in the first 70 epochs and 0.0001 in the last 50 epochs. 93
3.7 The total training loss trajectory of the proposed method M_4 over epochs using different learning rate schedules on the IEEE-118 bus system. The plot starts from the second epoch for better demonstration. 93

List of Tables

2.1 Hyperparameters of the NN structures. The second column indicates the number of neurons in each layer. The last column shows the size of each dataset. . . . 35
2.2 ARMSEs of voltage phasor calculations in different cases (10^{-4}) 37
2.3 ARMSEs of the branch flow calculations for different cases 37
2.4 AWDs of voltage phasor distributions in different cases (10^{-4}) 39
2.5 AWDs of branch flow distributions in different cases 39
2.6 The training time of each epoch for different NN structures (seconds). . 41
2.7 Hyperparameters of different methods on different bus systems 48
2.8 Average RMSEs of voltage magnitudes (10^{-4}) 50
2.9 Average RMSEs of voltage angles and angles differences calculations (10^{-3}) 50
2.10 Average RMSEs of branch flows calculations 50
2.11 AWD of the voltage phasors distributions (10^{-4}) 51
2.12 AWD of the branch flows distributions 52
2.13 Average MAE of estimate mean and std of branch flows 52

3.1 The nominal values of decision variables 64
3.2 Feasibility evaluation of the FDPF-Dual method 64
3.3 Performance comparison 66
3.4 Performance of the NGT method on the IEEE-30 bus system 67
3.5 Performance of the NGT method on the IEEE-118 bus system ($\tau = 0.5$) 68
3.6 The FCNN structure, weight parameters, training epochs, and learning rate. . . . 84
3.7 Methods for comparison 85
3.8 The average std values and the ridge regression parameters of the decision variables. . . . 87
3.9 The ℓ_2 norm of prediction errors of the ridge regression method. 87
3.10 Data preparation time. 87
3.11 The size of reduced branch set under different β_b. The third column shows the ratio of M_r to M. . . . 88

3.12 Comparison Results: Feasibility and Optimality. Training failure (✗) is indicated when the FDPF solver cannot find feasible PF solutions. The winners of methods without load mismatch are highlighted in bold. . . . 89
3.13 The training time and the memory size of the gradient data. 90
3.14 The testing time comparison results. 91

Abstract

Power Flow Analysis and Optimal Power Flow with

Physics-Informed Deep Learning

by

Kejun Chen

Power flow (PF) analysis is critical to power system operation and planning. Nowadays, renewable energy power generation has been widely installed in power grids because they are environmentally friendly. The high penetration of renewable energy brings significant fluctuations to the power system states. Probabilistic power flow (PPF) analysis aims to characterize the probability properties of voltage phasors with stochastic power injections.

Exploiting the impressive capability of neural networks (NNs) in complex function approximation, we utilize the NN as a rapid PF solver in real-time applications. Motivated by residual learning, the first work proposes a new NN structure based on the physical characteristics of PF equations. Specifically, we add a linear layer between the input and the output to the multilayer perceptron (MLP) structure. We design three schemes to initialize the NN weights for the shortcut connection layer based on the linearized PF equations. Numerical results show that the proposed approach outperforms existing NN frameworks in estimation accuracy and training convergence. However, the branch flow estimation accuracy of the NN-based methods on some benchmark systems is lower than the linearized PF-based method. The inherent reason is that the NN

outputs are voltage angles instead of voltage angle differences, while the latter determines the branch flows. To further improve the branch flow estimates, the second work separates the training of voltage magnitudes and phase angles due to their different properties. We incorporate the errors of voltage angle differences into the training loss function.

Based on PF equations, optimal power flow (OPF) analysis minimizes the total generation cost while subject to other operational constraints. To help the independent system operator (ISO) clear the real-time energy market, we develop an unsupervised learning-based framework to solve the OPF problem rapidly. We employ a modified augmented Lagrangian function as the training loss. The multipliers are updated dynamically during the training process based on the degree of constraint violation. Numerical results show that the dynamic updates of the penalty weight coefficient improve the feasibility of solutions compared to the fixed pre-assigned coefficient.

To ensure the PF balance, the NN predicts a subset of decision variables, and the remaining variables are obtained by a subsequent PF solver. However, the variable splitting scheme introduces heavy computation complexity when it comes to computing gradients in backpropagation. Hence, in the fourth work, we aim to reduce the total computational time of the NN to enable a daily update of the NN. We propose a physics-informed gradient estimation method based on a semi-supervised learning framework. We employ ridge regression to obtain pseudo-optimal solutions and build a hybrid dataset. We propose a batch-mean gradient estimation method based on the linearized Jacobian model to speed up the training process. Numerical results show

that the proposed gradient estimation method achieves a similar convergence rate as the ground truth Jacobian. Moreover, the proposed method rapidly obtains near-optimal solutions, which is appealing in real-time applications.

Acknowledgments

I sincerely thank my Ph.D. advisor, Professor Yu Zhang, for his time, patience, wisdom, expertise, dedication, efforts, and encouragement throughout my long Ph.D. journey, which contributed to a more solid and comprehensive study with his constructive and valuable advice. I highly appreciate his time, patience, support, and efforts in helping me review and revise my papers during my Ph.D. journey, which significantly improved the quality of this Ph.D. thesis and helped build the foundation of my research habit. I want to sincerely thank my committee members, Professor Patrick Mantey and Professor Leila Parsa, for their valuable advice, suggestions, and support in helping me improve the quality of this thesis. I would like to express my heartfelt gratitude to my labmates Jing Xiong, Sifat-E-Tanzim Chowdhury, Gabriel Intriago, and Shourya Bose. Their kindness, encouragement, and positive impact give me much support. I want to thank Baskin Engineering IT/Computing Office for the computational resources.

Additionally, this work was supported in part by the Faculty Research Grant of the University of California, Santa Cruz, and the Hellman Fellowship.

Chapter 1

Introduction

1.1 Overview

Power flow (PF) analysis determines the best operational points, which is essential for the power system operation and planning. PF studies learn the mapping from the input variables (load demands and power generations) to the output variables (voltage phasors and branch flows). Compared to conventional energy, such as fossil fuels, renewable energy tends to be much less harmful to the environment because renewable energy sources (RESs) do not produce greenhouse gas emissions [1]. However, renewable energy generation is generally sensitive to ambient conditions, such as pressure, temperature, humidity, and light intensity. This inherent randomness brings uncertainties to the operational states of the system, including voltage phasors (voltage magnitudes and phase angles) and branch flows (active and reactive branch flows). Therefore, the high penetration of renewable energy generation in electricity grids has presented more

challenges to system operators.

Probabilistic power flow (PPF) analysis describes the probability properties of the output variables given the uncertain input variables, which is essential in addressing system reliability and robustness. Monte Carlo simulations (MCS) methods and advanced MCS-based methods are prevalent in PPF analysis [2]. They generate the input data samples from stochastic power injection distributions and run PF analysis repeatedly to calculate their corresponding voltage phasors. Specifically, they employ PF solvers such as the Newton–Raphson (NR) method to solve the inverse alternating current PF (AC-PF) equations repeatedly [3]. Then, all the probability properties such as mean, standard deviation (std), and probability density function (PDF) of voltage phasors, can be obtained based on the data samples. However, a considerable amount of samples are necessary for accurate probability descriptions. Therefore, the cumulative computational time of MCS-based methods is heavy. One straightforward way to reduce the total computational time is to speed up the run time of the individual PF problem.

The widespread use of massive phasor measurement units (PMUs) has recently allowed abundant measurement data to be collected. Thus, machine learning-based approaches have gained increasing attention in PF analysis. For example, [4] and [5] use the linear regression method to approximate the decoupled linear PF functions. In addition, neural networks (NNs) have shown impressive capability in complex function approximation [6]. This inspires us to utilize the NN to solve the highly non-linear and non-convex alternating current PF (AC-PF) equations. The NN can serve as a PF

solver, which can achieve higher accuracy than traditional machine learning regression methods. The offline training phase is time-consuming due to the backpropagation process. Once the training is completed, the trained model can rapidly predict the corresponding voltage phasors of new input samples. The computational time of feed-forward propagation in the testing phase is low. The total computational time is still ignorable even if many samples are needed to obtain desirable probability descriptions. Therefore, PPF analysis can utilize the NN as the efficient workhorse of PF analysis.

We proposed two novel deep learning-based methods to approximate the inverse AC-PF mapping from power injections to voltage phasors. The main contribution of our first work is that we propose a new NN design under physical guidance to improve estimation accuracy and convergence rate [7]. The proposed NN structure based on residual learning, together with the well-designed initialization methods, achieves a faster convergence rate than the fully connected neural network (FCNN). We add a linear shortcut connection layer from the input to the output. Inspired by the linearized PF equations, we design three initialization schemes of the weights of the shortcut layer. Two model-based initialization methods require the power grid parameters, while the data-driven method requires the historical dataset. Compared to the random initialization scheme, the designed initialization methods achieve a faster convergence rate than the random initialization.

However, even if the voltage phasor estimates are accurate, the branch flow estimates on some test benchmark systems are inaccurate. The reason is that the NN outputs are phase angles instead of phase angle differences, while the latter determines

the branch flows. We combine multi-task learning and variation-cognizant voltage magnitudes to enhance branch flow estimation accuracy [8]. The training loss function consists of the voltage angle difference and the voltage angle estimate errors. In addition, we split the historical voltage magnitude data into two subsets based on the variation level of voltage magnitude at each bus. According to their std values, we use ordinary least-square linear regression to predict voltage magnitudes with smaller variations and the FCNN to estimate the larger ones. In the end, the proposed hybrid strategy improves the overall accuracy of voltage magnitude estimates.

AC optimal power flow (AC-OPF) analysis is another critical problem in electricity grids. It minimizes the objective function while satisfying physical constraints, including power balance and branch flow equations, and inequality constraints, such as box constraints of power generations, voltage magnitudes, and branch flows. AC-OPF problems are non-convex and highly non-linear due to the power balance equations. The non-linear programming (NLP) solvers can be used to solve AC-OPF problems. However, the computational time required to solve the AC-OPF problem for the large-scale bus system using the conventional optimization solver is long. Hence, the heavy computational burden makes them less appealing in real-time large system operations. Convex relaxation techniques can relieve the computational complexity, while inexact convex formulations may yield sub-optimal or infeasible solutions with significant optimality gaps [9]. Approximation methods such as the direct current OPF (DC-OPF) model linearize the AC-PF equations and significantly reduce the run time [10, 11]. However, due to the significant fluctuations in renewable energy generation and load demand, the

AC-OPF models are essential to prevent network losses that may arise from the low-fidelity linearized models. Besides, it is hard to recover the AC feasible solution from the solution obtained by the DC-OPF model [12]. Hence, there is a need to explore alternatives to conventional solvers, as they are unsuitable for real-time scenarios.

There is an increasing interest in applying deep learning-based approaches to AC-OPF analysis, driven by their notable function approximation capabilities. The offline-trained NN can serve as a rapid AC-OPF solver in the real-time energy market, which allows ISOs to obtain the optimal system states rapidly. Several variable splitting schemes have been proposed to reduce the violation of equality constraint. For example, [13], [14], and [15] use NNs to output voltage phasors and obtain the power generation values via power balance equations. To further address the load mismatch issue, [16] and [17] utilize NNs to predict the controllable decision variables of generator buses and reconstruct the other remaining decision variables by solving the PF equations. These supervised learning frameworks rely on conventional optimization solvers to generate data pairs, which requires extra data preparation time. Moreover, the resulting sub-optimal or infeasible solutions may mislead the NN training process. Hence, there is an increasing interest in employing unsupervised learning frameworks to bypass these drawbacks.

We propose an unsupervised end-to-end framework to predict a partial set of decision variables, including voltage phasors and power generations [18]. The nodal power balance equations are always satisfied by adopting the decision variables splitting scheme [19]. We use the augmented Lagrangian function as the training loss function,

which consists of the generation cost and the penalty term for constraint violation values. The Lagrangian multipliers are the weighting parameters in the training loss function. Compared to the fixed weighting parameters, the Lagrangian multipliers are adjusted dynamically according to the constraint violation degree, which improves the feasibility. In addition, the fast decoupled power flow (FDPF) solver is used to recover the voltage phasors [20], which reduces the training time compared to the NR method. The proposed unsupervised end-to-end framework can serve as a rapid AC-OPF solver, which is appealing for many real-time applications.

However, the variable splitting schemes introduce a heavy training burden when it comes to computing gradients for backpropagation. Specifically, the gradient of the reconstructed variables with respect to the NN output variables needs to be calculated at each training iteration. Ref. [19] shows that the gradient computation process involves the inverse operation of the Jacobian tensor according to the implicit function theorem. The long training time prevents frequently and promptly updating the NN with new data instances. In addition, without guidance from the data pairs, the unsupervised learning framework may suffer from training failure on large-scale bus systems in the worst scenario. The reason is that the subsequent FPDF solver cannot find any feasible PF solution given the initial random NN output decision variables.

Considering the data-driven nature of the NN, the NN needs to be updated timely by incorporating new data instances to ensure its performance. To have the data pairs for training guidance while not increasing the data preparation time, we propose a semi-supervised learning framework by utilizing data augmentation techniques to build

pseudo labels [21]. Instead of purely relying on the conventional optimization solver to generate data pairs, we first employ a conventional optimization solver to generate a small fraction of ground truth output variables for the input load demand samples and then use ridge regression to predict pseudo values for other unlabeled input samples. Due to the embedded PF solver, the inverse operation of the Jacobian matrix of reconstructed variables with respect to the NN output variables needs to be computed at each training iteration for backpropagation. Inspired by the linear AC-PF equations, we propose a new physics-informed gradient estimation method for the Jacobian tensor to alleviate the training burden and storage requirements [22]. The estimated gradients achieve comparable convergence performance as the ground truth gradients but require much less run time and storage. In addition, we remove branches (power lines) that are unlikely to violate their apparent branch flow limits during training to eliminate the unnecessary gradient computation for branch flow violation. This design is motivated by the observation that a significant number of branch flows are far from their operating limits [23, 24], and the resulting zero violation does not impact weight updates. The proposed semi-supervised learning framework can complete training in one day, which enables timely and frequent updates based on the latest data instances.

1.2 Literature Review

1.2.1 PPF analysis

The high penetration of renewable energy generation in the power grid brings significant fluctuations to the system operating states. PPF analysis focuses on obtaining the probability properties of voltage phasors under stochastic load demands and power generations [25]. The existing methods for PPF analysis generally fall into two categories: analytical and numerical methods [26].

Analytical methods attempt to obtain the output variables' PDFs given the input variables' PDFs. The output variable is represented by the linear combinations of input variables based on the first-order Taylor series expansion. This linear relationship may suffer significant errors when the input variables are far from nominal operating states. For example, convolution-based methods apply convolution operations to obtain the PDFs of output variables [27]. The computational burden is heavy due to the involvement of many convolution operations. Therefore, cumulants-based methods replace the convolution operations with arithmetic operations to reduce the computational complexity [28]. They calculate the self and joint cumulants of the output variables and use series expansions to estimate their PDFs. However, the number of joint cumulants that need to be calculated increases exponentially with more input variables and higher-order joint cumulants. Besides, various series expansions may have potential convergence failure issues. Moreover, using linear approximation equations leads to significant errors when power injections deviate far from the nominal operating

point [29].

In addition, point estimate methods can obtain the statistical moments of output variables based on the first few statistical moments of the input variables [30]. They respect the non-linear AC-PF equations and require less statistical information on the input variables than analytical methods. However, the estimation accuracy will decrease dramatically as the number of input variables increases, which leads to performance degradation in medium- and large-scale bus systems [31]. In addition, to handle random input variables inferred from data samples, [32] employs the non-linear partial least square method to de-correlate the random input variables and uses the arbitrary polynomial chaos expansion to obtain the probabilistic characteristics of the output variables. Refs. [33, 34] combine K-means clustering with cumulant to address the large fluctuations brought about by the high wind power integration. However, the performance of K-means clustering will degrade with the increasing number of renewable energy generators.

Given random power injection samples, numerical methods run deterministic PF analysis repeatedly to calculate their corresponding voltage phasors. Branch flows can be further determined using the branch flow equations when the voltage phasors are known. Finally, all probability properties of voltage phasors and branch flows are available. The MCS-based method, typically working with the NR solver, can accurately solve the inverse AC-PF equations. Some advanced MCS-based approaches, such as MCS with importance sampling [3] or Latin supercube sampling [35] and Quasi-Monte Carlo [36, 37] can improve the sample efficiency by selecting more representative data

samples. However, these MCS-based methods require many simulation runs to obtain accurate PDFs of the output variables. Due to the long cumulative computational time, MCS-based methods are less appealing in real-time applications in large electrical systems. Therefore, we focus on reducing the total computational time by speeding up the computation of individual PF analysis.

Recently, machine learning-based methods have gained more attention in PF analysis by shifting the training process offline. The mapping from the random input variables to the output variables can be directly learned from the historical operational data pairs. For example, [4] employs linear regression to learn the inverse AC-PF equations. However, using linear models to approximate the non-linear relationship may lead to significant errors. Ref. [38] uses the support vector regression method to approximate the non-linear AC-PF equations. Refs. [39] and [40] employ Gaussian process regression for PPF analysis. However, Gaussian process regression can only obtain the PDF of a single target quantity in one shot, which prevents its application in medium- and large-scale power systems if the PDFs of all buses' voltage phasors are required. In addition, NNs can approximate complicated functions according to the universal approximation theorem [6]. Therefore, [41] suggests using NNs to estimate the inverse AC-PF function. The NN's outputs are the normalized variables instead of the actual ones. It is hard to recover their original values and statistical properties precisely. Besides, the loss function consists of the mean square errors of active/reactive branch flows and voltage magnitudes/angles. Ref. [42] shows that it can be pretty challenging for NNs to optimize multiple tasks in one shot. Moreover, [43] employs NNs to solve the

inverse AC-PF equations, with an auxiliary task to rebuild the PF model. However, the NN's outputs are the real and imaginary parts of voltage phasors, and converting them to voltage magnitudes and angles may magnify the prediction errors. To exploit the power grid topology features, [44] proposes a physical-guided graph neural network that incorporates the physics of PF equations. However, the performance of the proposed model is highly affected by the accuracy of topology information, which decreases the model's generalization ability under varying grid topologies [45].

Inspired by previous work, we proposed two novel deep learning-based approaches for rapid PF analysis in real-time scenarios. Motivated by residual learning, the first work proposes a new physics-guided NN structure to approximate the non-linear residuals of the inverse AC-PF equations. The main contributions are two-fold:

- We introduce a linear layer between the input and output in the MLP structure. The shortcut connection layer captures the linearity of AC-PF equations while the MLP learns the non-linear residuals.

- We propose three initialization schemes to initialize the weights of the shortcut connection layer; two model-based schemes require the grid topology and line parameters, while the data-driven-based initialization method can work without these parameters.

The proposed framework has the voltage angles as the NN outputs instead of the voltage angle differences. However, the branch power flows are directly related to voltage angle differences instead of the voltage angles. We find the model perfor-

mance degrades when it comes to computing the branch flow estimates. In the second work, we employ two different data-driven methods to further improve voltage magnitude estimates and multi-task learning to enhance branch flow estimates. The main contributions of the second work can be summarized as follows:

- We utilize two different NNs to separate the training of voltage magnitudes and phase angles because they have different features.
- We split load buses into two sub-datasets based on historical variations of voltage magnitudes. For the voltage magnitude prediction, we use simple ordinary least-square linear regression instead of the NN for the load bus whose voltage magnitude has relatively small variations.
- Branch flows are more related to voltage angle differences than voltage angles. Therefore, we use the joint loss of voltage angles and voltage angle differences to help further improve the estimation accuracy of branch flows.

1.2.2 AC-OPF analysis

AC-OPF analysis is essential for power system operation and planning in power systems. It minimizes the total power generation cost while satisfying many operational constraints, such as power balance equations, power generation limits, and branch flow limits. AC-OPF problems can be solved by NLP solvers, such as the interior point method. However, due to the highly non-linear power balance equations, solving the non-convex AC-OPF problem by NLP solvers can be computationally expensive [46, 47].

Hence, there is a need to explore rapid AC-OPF solvers for real-time scenarios. The existing approaches for AC-OPF analysis generally fall into three categories: convexification, approximation, and data-driven methods.

Convex relaxation refers to transforming a non-convex optimization problem into a convex one. Various convex relaxations have been applied to the AC-OPF problems to provide tighter objective value bounds and improve the optimality gaps. For example, semi-definite programming (SDP) relaxations can recover the exact solution to the original AC-OPF problem under certain conditions [48, 49]. Second-order cone programming (SOCP) relaxations are more scalable to large-scale power grids due to their advantages in computation efficiency over SDP relaxations [50, 51]. However, SOCP relaxations do not retain the phase angle information, which prevents their usage in a transmission power grid with mesh topology. Quadratic convex (QC) relaxations enclose the sine and cosine functions in the AC-PF equations to construct convex envelopes [52, 53]. Ref. [9] shows that these relaxed problems can provide lower bounds for the original AC-OPF problem and achieve a zero-duality gap under certain conditions. However, if convex relaxations are inexact, e.g., the rank-1 constraint violation [54], they provably yield AC infeasible solutions.

Approximation methods adopt linearized PF equations in the AC-OPF problem formulation [10, 11]. With a desirable approximation method, the obtained solution should be close to the optimum of the original problem. One of the most popular approaches is DC-OPF analysis, which assumes flat voltage magnitudes and ignores line losses and reactive power injections. The DC-OPF problem can be solved quickly for

large-scale power grids; thus, they have been widely used for real-time market clearing. However, it is challenging to recover an AC-feasible solution from the DC-OPF solution [12].

Many research efforts have been made to utilize deep learning-based approaches to help conventional solvers conduct rapid AC OPF analysis. Ref. [55] employs a classification algorithm to tell apart the active and inactive constraints and then reduces the optimization problem size by removing those inactive constraints. Ref. [56] uses NN to provide warm-start points for the conventional optimization solvers to speed up their computational time. Driven by the powerful approximation capabilities of NNs, many approaches focus on an end-to-end NN framework to learn the optimal mapping directly. The NN takes in the active and reactive power demand and outputs the power generation and voltage phasors. The time-consuming training process is shifted offline. In the real-time energy market, the ISO can employ the trained NN to rapidly solve the AC-OPF problem and obtain the optimal system operating state. These approaches can be divided into two main categories: supervised learning and unsupervised learning.

Based on historical data pairs, the NN used in [57] outputs all decision variables simultaneously but ignores equality and inequality constraint violations, which may cause significant load mismatches and inequality constraint violations. Ref. [58] incorporates inequality constraint violation penalty in the training loss function, which helps to obtain the decision variables that satisfy the inequality constraints. However, it does not consider the power balance equations and may lead to load mismatches. For enhancing solution feasibility, there is an increasing research interest in using NNs

to predict a partial set of decision variables and reconstruct the remaining variables through equality constraint completion. There are generally two different schemes to split decision variables. For example, [14] and [15] use the FCNN to predict the voltage magnitudes and phase angles and then compute active and reactive power generations using power balance equations. Moreover, graph neural networks, convolutional neural networks (CNNs), and Chebyshev CNNs can take the place of FCNNs to leverage the power grid's topology information; see [57, 59, 60].

Ref. [61] employs principal component analysis to compress the space of output decision variables to speed up the training efficiency. However, using the NN to output all the decision variables simultaneously cannot strictly satisfy equality constraints. For example, these proposed methods cannot ensure the power balance equation is satisfied with load buses. Therefore, [16] and [17] use the NN to output the active power generations and voltage magnitudes of generator buses and the voltage magnitude of the slack bus. The remaining decision variables can be obtained by solving AC-PF equations to satisfy power balance equations. To this end, supervised learning-based methods typically rely on conventional solvers to generate training data pairs that serve as ground truth. It may take a long time for conventional optimization solvers to generate many samples for a large-scale bus system. Moreover, the resulting solutions may not be global optimum and can mislead the NN training process.

Semi-supervised learning can cope with the missing or imbalanced dataset issue and has been widely applied in detecting abnormal events and faults [62, 63]. For AC-OPF analysis, iteration-based optimization solvers are typically utilized to obtain the

optimal targets, which may lead to long run time in large-scale bus systems. Therefore, there is increasing interest in developing unsupervised learning frameworks for solving the AC-OPF problems without the aid of conventional optimization solvers. The feasibility and optimality can be ensured by designing a proper training loss function, including the generation costs, equality constraint violations penalties, and inequality constraint violations penalties [64, 65]. To ensure load demand balance, [19] uses the NN to predict the controllable decision variables and employs a PF solver to obtain the voltage magnitude and phase angles. However, balancing the weighting parameters of different loss terms can be challenging in multi-task learning. For example, increasing the weighting parameter of the generation costs term may promote solutions with lower costs but larger constraint violations. Motivated by the principle of augmented Lagrangian, ref. [66] trains the primal and dual NNs jointly and uses the estimated dual values for the training loss function of the primal NN. In this case, inaccurate dual variable estimates in the initial training process will mislead the primal NN learning direction.

Inspired by previous work, we propose an end-to-end unsupervised learning framework for AC-OPF analysis. The equality constraints are guaranteed to be satisfied via the PF solver. The contributions can be summarized as follows:

- We employ the augmented Lagrangian function as the training loss function, including the total generation cost and inequality constraints violation value. The Lagrangian multipliers are adjusted dynamically according to the inequality constraints violation value during training. Compared to the fixed weight parameter,

the proposed dynamic parameter adjustment helps improve the feasibility of the solution.

- We employ FDPF to solve the AC-PF equations, which significantly reduces the computational time in the forward propagation compared to the NR method.

However, integrating the PF solver into the learning framework brings new challenges to the NN training process regarding computing gradients in backpropagation. Specifically, the gradients of the recovered decision variables with respect to the NN output variables need to be calculated at each training iteration. Ref. [19] calculates the ground truth Jacobian of the AC-PF equations based on the implicit function theorem. However, the computational complexity of computing the inverse of the Jacobian grows quadratically with the power grid size, which leads to a long training time. Due to the data-driven nature, frequent updates of the NN by accommodating new data instances are essential to ensure that the quality of NN solutions does not degrade [67]. To alleviate the heavy computational burden, [16] and [17] adopt the zeroth-order gradient estimation method [5]. The zeroth-order gradient estimation method is easier to implement than the ground truth Jacobian. However, the resulting gradients can be far from the ground truth values, which may provide an incorrect gradient descent direction and mislead the NN learning process. Moreover, without data pair guidance, the subsequent PF solver in the unsupervised learning framework may fail to find any feasible solution in the initial training process.

We propose novel physics-informed gradient estimation methods based on a

semi-supervised learning framework to improve the convergence property. The proposed method can achieve timely and frequent updates of the NNs for large-scale bus systems. The contributions of the fourth work can be summarized as follows:

- We use ridge regression to build a hybrid dataset instead of relying purely on the conventional optimization solver to generate data pairs. Ridge regression can rapidly predict pseudo-optimal solutions for the given input load demands. The pseudo-pairs can provide practical guidance for the NN learning process and improve the convergence rate.

- We exclude power lines that are unlikely to be binding to their flow limits to curtail unnecessary gradient computations. In addition, we develop physicals-informed batch-mean gradient estimation methods to accelerate the gradient computation in the backpropagation process. The proposed gradient method achieves comparable performance as the ground truth Jacobian while the computation complexity is significantly reduced, which enables a frequent and daily NN update.

The rest of the thesis is organized as follows. Chapter 2 formulates the PF problem and connects PF analysis and PPF analysis. In addition, the residual learning-based NN framework and the multi-task learning framework are detailed. Chapter 3 formulates the AC-OPF problem and describes unsupervised and semi-supervised learning frameworks. The conclusion and future work are depicted in Chapter 4.

Notation: Upper (lower) boldface letters are used for matrices (column vectors). Sets are denoted by calligraphic letters. $(\cdot)^\top$ is vector/matrix transpose; $\|\cdot\|_2$

denotes vector ℓ_2-norm; $(\cdot)^{-1}$ and $(\cdot)^{\dagger}$ denote inverse and pseudo-inverse, respectively. $\mathbf{1}_n \in \mathbb{R}^n$ is the all-ones column vector. $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ is the identity matrix of size n. $\odot$ denotes the element-wise product.

1.3 Contributions

The main contributions of this thesis can be summarized as:

1. Inspired by residual learning, we propose a novel physics-guided NN framework to solve the AC-PF equations. Leveraging the linearized approximation of the AC-PF equations, we propose three schemes to initialize the weights of the shortcut connection layer. It can be employed as a rapid PF solver to reduce the cumulative computational time of lots of samples in PPF analysis.

2. In PF analysis, we develop a hybrid learning strategy based on the variation level of the voltage magnitude at each bus to improve the estimation accuracy. In addition, we utilize the multi-task learning technique to improve the branch flow estimation accuracy by incorporating the prediction errors of voltage angle differences into the training loss function.

3. We propose an end-to-end unsupervised learning framework for AC-OPF analysis. We integrate the PF solver into the learning framework to satisfy the AC-PF equations. In addition, we utilize the augmented Lagrangian function as the training loss, where the Lagrange multipliers are adjusted adaptively to improve the feasibility of the solution.

4. In AC-OPF analysis, we adopt a semi-supervised learning framework aided by the pseudo-labeling technique to provide sufficient guidance for the NN updates while not introducing extra long data pair preparation time. To reduce the NN training time, we propose physics-informed batch-mean gradient estimation approaches along with a reduced branch set to relieve the complexity of the gradient computation.

Chapter 2

Probabilistic Power Flow Analysis

This section describes the proposed two novel deep learning-based frameworks for PPF analysis. The proposed NN can serve as a rapid AC-PF solver and reduce the total computation time for many data samples in PPF analysis. Inspired by residual learning, the first work designs a new physics-informed neural network framework to improve the voltage phasor estimation accuracy and NN training convergence rate. However, the advantages of the proposed learning framework are not evident in the branch flow estimates compared to the voltage phasor estimates. The potential reasons are that the NN outputs are phase angles instead of phase angle differences, while the latter determines the branch flows. Hence, the second work proposes a variation-cognizant strategy to improve the voltage magnitude estimate accuracy and utilizes multi-task learning to improve branch flow accuracy.

2.1 Problem Formulation

PF analysis is the cornerstone of PPF analysis. Hence, we introduce the power system description and the AC-PF problem formulation and then connect the PF problem to the PPF analysis.

2.1.1 System Description

Given the nodal power injections, PF analysis attempts to obtain the system operating status in an electrical grid, including voltage magnitudes and phase angles. There are three different types of buses in the power system modeling: PQ buses, PV buses, and one slack bus. A PQ bus (a.k.a. load bus) has no generator connected, where its active and reactive power injections are specified. A PV bus (a.k.a. generator bus) has generators connected, and its active power and voltage magnitude are given. Lastly, the slack bus has the specified voltage angle and magnitude. Therefore, the phase angles and voltage magnitudes of PQ buses and voltage angles of PV buses are unknown system status. Here, consider an electrical grid with M transmission lines and N buses, where there are N_g PV buses (denoted by set $\mathcal{N}_g$), $N - N_g - 1$ PQ buses (denoted by set $\mathcal{N}_l$), and one slack bus.

2.1.2 AC-PF Problem Formulation

According to Kirchhoff's Law, the AC-PF equations can be expressed as:

$$P_i = \sum_{j=1}^{N} V_i V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}),\ \forall i \in \mathcal{N}_g \cup \mathcal{N}_l\,, \tag{2.1a}$$

$$Q_i = \sum_{j=1}^{N} V_i V_j (G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij}),\ \forall i \in \mathcal{N}_l\,, \tag{2.1b}$$

where P_i and Q_i are the active and reactive power injections at bus i. V_i and θ_i are the corresponding voltage magnitude and phase angle. $\theta_{ij} := \theta_i - \theta_j$ is the angle difference between bus i and j. G_{ij} and B_{ij} are the real and imaginary parts of the (i,j)-th element of the nodal admittance matrix $\mathbf{Y} \in \mathbb{C}^{N\times N}$. The active and reactive branch flows between the connected buses i and j can be calculated by:

$$P_{ij} = \frac{V_i V_j}{t_{ij}} (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}) - \frac{G_{ij} V_i^2}{t_{ij}^2}\,, \tag{2.2a}$$

$$Q_{ij} = \frac{V_i V_j}{t_{ij}} (G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij}) + \frac{B_{ij}}{t_{ij}^2} V_i^2 - \frac{b_{ij}^c}{2t_{ij}^2} V_i^2\,, \tag{2.2b}$$

where t_{ij} and b_{ij}^c denote the tap ratio and the total line-charging susceptance between bus i and j, respectively.

Let $\mathbf{z}$ collect voltage magnitudes and phase angles of all buses:

$$\mathbf{z} := [\theta_s; \boldsymbol{\theta}_g; \boldsymbol{\theta}_l; V_s; \mathbf{V}_g; \mathbf{V}_l]^\top\,, \tag{2.3}$$

where subscripts $(\cdot)_s$, $(\cdot)_g$ and $(\cdot)_l$ denote the quantities corresponding to the slack

bus, generator buses, and load buses, respectively. Let $\mathbf{x} = [\mathbf{P}_g; \mathbf{P}_l; \mathbf{Q}_l]^\top$ and $\mathbf{y} = [\boldsymbol{\theta}_g; \boldsymbol{\theta}_l; \mathbf{V}_l]^\top$. The nodal admittance matrix $\mathbf{Y}$ can be partitioned similarly, and it can be reorganized to a new matrix $\tilde{\mathbf{Y}}$ given in Eq.(2.4).

$$\tilde{\mathbf{Y}} = \begin{bmatrix} \mathbf{Y}_{ss} & \mathbf{Y}_{sg} & \mathbf{Y}_{sl} \\ \mathbf{Y}_{gs} & \mathbf{Y}_{gg} & \mathbf{Y}_{gl} \\ \mathbf{Y}_{ls} & \mathbf{Y}_{lg} & \mathbf{Y}_{ll} \end{bmatrix}, \tag{2.4}$$

where, for instance, $\mathbf{Y}_{gs}$ is formed from rows of $\mathbf{Y}$ that correspond to PV buses and the column for the slack bus. To this end, the inverse AC-PF function, denoted as $\mathbf{f}(\cdot)$, can be compactly expressed as (2.5). Based on the AC-PF equations (2.1), the PF solver aims to calculate the unknown voltage phasors in $\mathbf{y}$ for the given nodal power injections in $\mathbf{x}$.

$$\mathbf{y} = \mathbf{f}(\mathbf{x})\,. \tag{2.5}$$

2.1.3 From PF Analysis to PPF Analysis

PPF analysis attempts to characterize the uncertainties of voltage phasors brought by the fluctuations of power injections. MCS methods have been widely applied to PPF analysis. Given the power injection samples, MCS methods repeatedly conduct PF analysis to calculate the corresponding voltage phasor for all samples. The probabilistic descriptions of voltage phasors can be further obtained. A considerable number of samples are required to guarantee accurate probability distribution. There-

fore, the total computational burden is heavy. In this case, speeding up the computation of PF analysis can help reduce the cumulative computation time of PPF analysis.

Deep learning techniques have gained increasing attention for PF analysis because the NN has shown powerful generalization abilities in function approximation. The NN takes in the power injections and predicts the corresponding voltage phasors. The NN can be employed as a rapid PF solver by shifting the time-consuming training process offline. The reasons are as follows. There are typically two stages of the NN-based methods: training and testing. In the training phase, the NN weights are updated during backpropagation to minimize the prediction errors of the voltage phasors. The training process may be time-consuming due to backward propagation. In the testing phase, a new power injection sample is fed into the trained NN, and the corresponding voltage phasors can be predicted rapidly. The computation time in the testing stage is ignorable because the computational burden of the forward propagation is low. To this end, the trained NN can rapidly obtain the phase angles for many power injection samples in PPF analysis.

2.2 Physics-guided Residual Learning for PPF Analysis

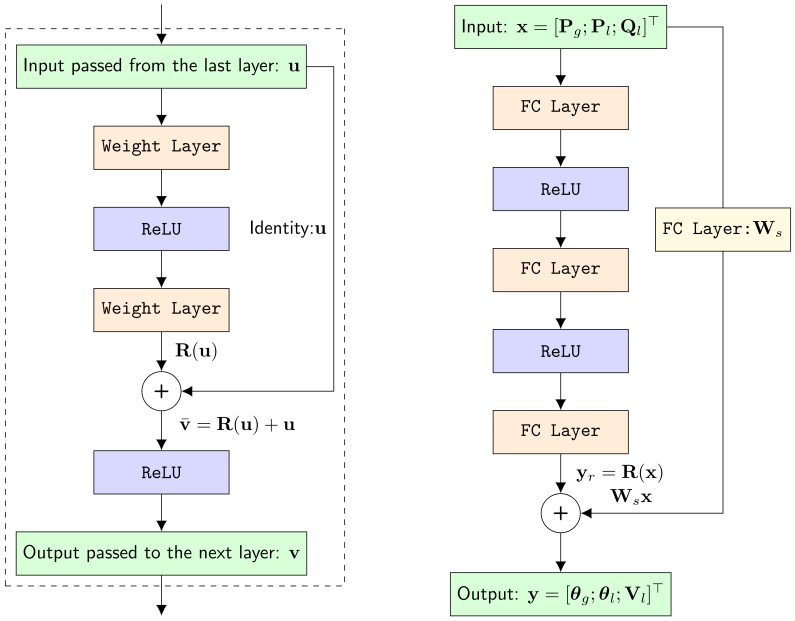
In this section, we propose an NN framework based on the physical characteristics of the AC-PF equations. First, we design a new NN structure motivated by residual learning. Specifically, we add a linear layer from the input to the output. Then, we propose three initialization schemes for the short connection layer based on the linearity

of the AC-PF equations. Two model-based schemes require line parameters and grid topology information. Under the missing or inaccurate power line parameters scenario, we design a data-driven scheme that only needs historical data samples.

2.2.1 Designed Framework inspired by Residual Learning

Residual neural networks (ResNets) have been widely applied to image recognition. Based on the vanilla MLP structure, ResNets introduce some extra shortcut connection layers [68]. A representative residual block that skips two weight layers is depicted in Fig. 2.1a. Let $\mathbf{G(\cdot)}$ denote the underlying mapping $\mathbf{u} \mapsto \bar{\mathbf{v}}$. The MLP is used to approximate the residual function $\mathbf{R}(\cdot)$. In this case, the original function $\mathbf{G(u)} := \mathbf{R(u)} + \mathbf{u}$ can be achieved by introducing a shortcut connection layer. Compared to vanilla MLPs, ResNets can effectively address the accuracy degradation issue, i.e., the NN performance decreases with the number of layers increasing.

We design an NN structure by adding a shortcut connection layer between the input and the output, as shown in Fig. 2.1b. Instead of learning the non-linear mapping $\mathbf{x} \mapsto \mathbf{y}$ directly, the designed structure uses the MLP to approximate the latent residual function $\mathbf{R(\cdot)}$. In the context of PF analysis, this implies the MLP aims to learn the non-linearity lying in the AC-PF equations while the linearity is captured via the linear shortcut connection layer. The motivations are given as follows. The linearized PF equations are widely adopted in PF analysis for the transmission power grid due to the physics of AC-PF equations. Thus, the linear shortcut connection layer is capable of learning the linearity. Supposing the underlying mapping is close to an identity function,



(a) One residual building block skipping two layers

(b) Proposed NN structure inspired by residual learning.

Figure 2.1: The residual block structure and the proposed NN structure based on residual learning.

[68] claims that pushing the residual to zero is likely easier than directly learning the desired function. Inspired by this principle, we consider learning the residuals regarding the linearized AC-PF equations easier than directly learning the non-linear function. The MLP structure comprises fully connected linear layers and rectified linear unit (ReLU) activation functions, denoted as $\boldsymbol{\sigma}$. Let $\mathbf{W}_i$ and $\mathbf{b}_i$ represent the weight matrix and bias of the i-th fully connected linear layer. $\mathbf{W}_s$ and $\mathbf{b}_s$ denote the weight matrix and bias of the shortcut connection layer. Formally, as shown in Fig. 2.1b, the residual output $\mathbf{y}_r$ and the output $\mathbf{y}$ can be expressed as:

$$\mathbf{y}_r = \mathbf{W}_3(\boldsymbol{\sigma}(\mathbf{W}_2(\boldsymbol{\sigma}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)) + \mathbf{b}_2)) + \mathbf{b}_3\,, \tag{2.6}$$

$$\mathbf{y} = \mathbf{y}_r + (\mathbf{W}_s\mathbf{x} + \mathbf{b}_s)\,. \tag{2.7}$$

2.2.2 Initialization Schemes for the Shortcut Layer

This section introduces three initialization schemes of the linear shortcut connection layer. The motivations are given as follows. If $\mathbf{W}_s$ and $\mathbf{b}_s$ are initialized randomly, the value of $\mathbf{y}_r = \mathbf{y} - (\mathbf{W}_s\mathbf{x} + \mathbf{b}_s)$ is prone to be updated randomly in the initial training stage. However, the advantage of residual learning will be more evident if the residual $\mathbf{y}_r$ can be pushed to close zero, which speeds up the learning process. To drive $\mathbf{y}_r$ to zero, $\mathbf{W}_s\mathbf{x} + \mathbf{b}_s$ should be close to the output $\mathbf{y}$. Based on the linear property of the AC-PF function, three schemes are adopted to initialize the shortcut linear connection layer, including two model-based methods and one data-driven-based method.

2.2.2.1 Pre-Initialization using a Linearized PF model

A decoupled linearized PF model has been proposed in [69]:

$$P_i = \sum_{j=1}^{N} -B'_{ij}\theta_j + \sum_{j=1}^{N} G_{ij}V_j,\ i \in \mathcal{N}_g \cup \mathcal{N}_l\,, \tag{2.8a}$$

$$Q_i = \sum_{j=1}^{N} -G_{ij}\theta_j + \sum_{j=1}^{N} -B_{ij}V_j,\ i \in \mathcal{N}_l\,, \tag{2.8b}$$

where B'_{ij} denotes the imaginary part of the (i,j)-th element of the nodal admittance matrix without the shunt elements, line-charging susceptance, as well as the equivalent admittance of transformers. If we separate the unknown voltage magnitudes and phase angles with known ones, Eq.(2.8) can be rewritten as:

$$\mathbf{x} = \mathbf{E}\mathbf{c} + \mathbf{F}\mathbf{y}\,, \tag{2.9}$$

where

$$\mathbf{E} = \begin{bmatrix} -\mathbf{B}'_{gs} & \mathbf{G}_{gs} & \mathbf{G}_{gg} \\ -\mathbf{B}'_{ls} & \mathbf{G}_{ls} & \mathbf{G}_{lg} \\ -\mathbf{G}_{ls} & -\mathbf{B}_{ls} & -\mathbf{B}_{lg} \end{bmatrix} \in \mathbb{R}^{(2N-N_g-2)\times(N_g+2)}\,,$$

$$\mathbf{F} = \begin{bmatrix} -\mathbf{B}'_{gg} & -\mathbf{B}'_{gl} & \mathbf{G}_{gl} \\ -\mathbf{B}'_{lg} & -\mathbf{B}'_{ll} & \mathbf{G}_{ll} \\ -\mathbf{G}_{lg} & -\mathbf{G}_{ll} & -\mathbf{B}_{ll} \end{bmatrix} \in \mathbb{R}^{(2N-N_g-2)\times(2N-N_g-2)},$$

$$\mathbf{c} = [\theta_s; V_s; \mathbf{V}_g]^\top \in \mathbb{R}^{N_g+2}\,.$$

Matrices $\mathbf{E}$ and $\mathbf{F}$ are determined by the power grid topology and line parameters. In addition, $\mathbf{c}$ collects the voltage phasor of the slack bus and voltage magnitudes of the PV buses. Based on Eq.(2.9), the linearized mapping from $\mathbf{x}$ to $\mathbf{y}$ can be calculated by:

$$\mathbf{y} = \mathbf{F}^\dagger\mathbf{x} - \mathbf{F}^\dagger\mathbf{E}\mathbf{c}\,, \tag{2.10}$$

where $\mathbf{F}^{\dagger}$ denotes the pseudo-inverse of $\mathbf{F}$. Note that $\mathbf{F}$ may not be invertible due to the possible zero bus injections. To this end, we can use $\mathbf{F}^{\dagger}$ and $-\mathbf{F}^{\dagger}\mathbf{Ec}$ to pre-initialize $\mathbf{W}_s$ and $\mathbf{b}_s$, respectively.

2.2.2.2 Pre-Initialization using Jacobian model

The first-order Taylor series expansion of AC-PF equations around the nominal operating point, denoted as $(\mathbf{x}_0, \mathbf{y}_0)$, can be expressed as:

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{J}(\mathbf{y} - \mathbf{y}_0)\,, \tag{2.11}$$

$$\mathbf{y} = \mathbf{J}^{-1}\mathbf{x} - \mathbf{J}^{-1}\mathbf{x}_0 + \mathbf{y}_0\,, \tag{2.12}$$

where $\mathbf{J} \in \mathbb{R}^{(2N-N_g-2)\times(2N-N_g-2)}$ refers to the Jacobian matrix evaluated at the nominal operating point. We can use $\mathbf{J}^{-1}$ and $-\mathbf{J}^{-1}\mathbf{y}_0 + \mathbf{x}_0$ to pre-initialize $\mathbf{W}_s$ and $\mathbf{b}_s$ so that the residual output represents the higher-order remainder of the Taylor series in the first training iteration. Given that the power injection values are close to the nominal values, the residuals are expected to be close to zero, which helps speed up the NN training process.

2.2.2.3 Pre-Initialization using Data-driven model

The aforementioned physics-guided initialization methods may not work due to missing or inaccurate information on power grid topology and line parameters. We propose a data-driven-based initialization scheme using ridge regression, which only

requires the historical dataset. The advantages of using ridge regression over ordinary least-square regression are as follows. Ref. [70] shows that significant initial values of the NN weights may cause the gradient explosion problem. The regularization term in ridge regression can effectively shrink the coefficients compared to the ordinary linear regression method, which contributes to obtaining NN initial weights with small values.

Let $\mathbf{X} \in \mathbb{R}^{n\times(2N-N_g-2)}$ and $\mathbf{y}_i^o \in \mathbb{R}^{n\times 1}$ denote the input data and the i-th dimension of the output data, where n denotes the number of training samples. Ridge regression aims to find the optimal value of $\boldsymbol{\beta}_i$ and α_i by solving the problem:

$$\underset{\boldsymbol{\beta}_i,\alpha_i}{\arg\min}\ L := \|\mathbf{y}_i^o - (\mathbf{X}\boldsymbol{\beta}_i + \alpha_i \mathbf{1}_n)\|_2^2 + \lambda_r \|\boldsymbol{\beta}_i\|_2^2 \,. \tag{2.13}$$

$\boldsymbol{\beta}_i^\top$ and α_i can be used to initialize the i-th row of $\mathbf{W}_s$ and $\mathbf{b}_s$, respectively. λ_r is a positive parameter that balances the emphasis given to minimizing the fitting error of data pairs vs minimizing the regularization term of coefficients. If $\lambda_r = 0$, ridge regression will degrade to the ordinary least-square linear regression model.

The closed-form solution to Eq.(2.13) is derived as follows. The objective function L can be rewritten as:

$$L = \boldsymbol{\beta}_i^\top(\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I}_{2N-N_g-2})\boldsymbol{\beta}_i - 2\boldsymbol{\beta}_i^\top\mathbf{X}^\top\mathbf{y}_i^o + 2\alpha_i\boldsymbol{\beta}_i^\top\mathbf{X}^\top\mathbf{1}_n - 2\alpha_i\mathbf{1}_n^\top\mathbf{y}_i^o + \alpha_i^2\mathbf{1}_n^\top\mathbf{1}_n + \mathbf{y}_i^{o\top}\mathbf{y}_i^o \,. \tag{2.14}$$

The gradients of the objective function with respect to the parameters can be expressed

as:

$$\nabla_{\boldsymbol{\beta}_i} L = -2(\mathbf{X}^\top \mathbf{y}_i^o - \mathbf{X}^\top \mathbf{X}\boldsymbol{\beta}_i - \alpha_i \mathbf{X}^\top \mathbf{1}_n - \lambda \boldsymbol{\beta}_i)\,, \tag{2.15}$$

$$\nabla_{\alpha_i} L = -2\mathbf{1}_n^\top \mathbf{y}_i^o + 2\mathbf{1}_n^\top \mathbf{X}\boldsymbol{\beta}_i + 2\alpha_i \mathbf{1}_n^\top \mathbf{1}_n\,. \tag{2.16}$$

We have $\nabla_{\boldsymbol{\beta}_i} L = \mathbf{0}$ and $\nabla_{\alpha_i} L = 0$ at the optimal solution. Hence, the optimal values of $\boldsymbol{\beta}_i$ and α_i should satisfy:

$$\boldsymbol{\beta}_i^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_{2N-N_g-2})^{-1}(\mathbf{X}^\top \mathbf{y}_i^o - \alpha_i^* \mathbf{X}^\top \mathbf{1}_n)\,, \tag{2.17}$$

$$\alpha_i^* = \frac{1}{n}(\mathbf{1}_n^\top \mathbf{y}_i^o - \mathbf{1}_n^\top \mathbf{X}\boldsymbol{\beta}_i^*)\,. \tag{2.18}$$

Based on Eqs.(2.17) and (2.18), we can further calculate $\boldsymbol{\beta}_i^*$ by:

$$\boldsymbol{\beta}_i^* = \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_{2N-N_g-2} - \mathbf{X}^\top \mathbf{H}\mathbf{X}\right)^{-1} \mathbf{X}^\top (\mathbf{I}_{2N-N_g-2} - \mathbf{H})\mathbf{y}_i^o \tag{2.19}$$

with $\mathbf{H} = \frac{1}{n}\mathbf{1}_n \mathbf{1}_n^\top \in \mathbb{R}^{n\times n}$. α_i^* can be obtained by plugging Eq.(2.19) to Eq.(2.18).

2.2.3 Simulation Results

The effectiveness of our proposed methods is verified based on the IEEE-30, IEEE-118, IEEE-300 [71], SouthCarolina-500 [72], and PEGASE-1354 [73] bus systems.

2.2.3.1 Data generation

We use a mixture of synthetic and real datasets for a comprehensive evaluation. The active load demands use the actual data provided by the global energy forecasting competition 2012 [74]. Then, a power factor is drawn randomly from the uniform distribution over the interval $(0, 1)$. The values of data samples are scaled to match the power system capacity. For the IEEE-30 bus, the active power generations are modeled as multivariate Gaussian distributions; the ratio of the std value to the mean value is 0.2. The correlation coefficient among different buses is 0.2. On the IEEE-118 and IEEE-300 bus systems, active power generations are from actual PV plants installed in California [75]. The load demands are modeled as multivariate Gaussian distributions. The ratios of the std values to the mean values are 0.1, 0.01, and 0.2 for the IEEE-118, IEEE-300, and SouthCarolina-500 bus systems, respectively [76]. The correlation coefficients for the active and reactive load demands are 0.8 and 0.2 between the same bus and different buses, respectively. For the PEGASE-1354 bus system, the ratio of the std value to the mean value is 0.1 for the active power generations and load demands. Finally, we generate training data pairs using the NR solver in `Matpower 7.0` [71].

2.2.3.2 Methods for comparison

We compare the proposed methods with the following five existing works:

- *Cumulants* [28]: Based on the first-order Taylor expansion, AC-PF equations are linearized around the nominal operating point (cf. Eq.(2.12)).

- *FC* [41]: The model-based initialization method and activation function are designed based on the MLP structure.

- *TPBNN* [43]: The MLP is adopted to approximate the inverse AC-PF function, with an auxiliary task to rebuild the forward AC-PF mapping. The reconstruction error serves as a regularization term in the training loss function.

- *ResNet* [68]: The ResNet is composed of stacked residual building blocks. The identity mapping in the residual block (cf. Fig. 2.1a) is replaced with a linear layer to improve its generalization capability. The output layer is linear due to possible negative values of voltage angles.

- *RR*: Ridge regression is used to learn the inverse AC-PF equations (cf. (2.13)). λ_r is set to 10^{-4} in the simulations.

In addition, we adopt four different initialization schemes for the shortcut connection linear layer, including random [77], data-driven PF model, linearized PF model, and Jacobian model, to initialize the shortcut connection linear layer of our proposed framework. They are named as *Random*, *Data-driven*, *Linearized PF*, and *Jacobian*, respectively.

2.2.3.3 Training details

We use the Adam optimizer with mini-batch for the NN training [78]. The batch size is 32, and the training loss function is the mean square error (MSE). The validation dataset is used to tune the hyperparameters. The training process will stop

when the validation loss has no further improvement [79]. We train and test five times to alleviate randomness. Table 2.1 shows the neural network structures and the dataset size tested on the benchmark systems. The simulations are implemented on an iMac equipped with an i7-8007 CPU and 32GB RAM. The neural network training is based on `PyTorch 1.7.1` in `Python 3.7`.

Table 2.1: Hyperparameters of the NN structures. The second column indicates the number of neurons in each layer. The last column shows the size of each dataset.

Cases	Structure	[Training, Validation, Testing]
30	[53 100 100 53]	[12k, 4k, 4k]
118	[181 300 300 181]	[20k, 5k, 5k]
300	[530 200 200 200 530]	[20k, 5k, 5k]
500	[909 300 300 300 300 909]	[28k, 6k, 6k]
1354	[2447 300 300 300 300 2447]	[34k, 8k, 8k]

2.2.3.4 Evaluation criteria

We adopt three different evaluation criteria for comprehensive evaluation:

- Average root mean square error (ARMSE): Let $\mathbf{O}$, $\hat{\mathbf{O}} \in \mathbb{R}^{M\times D}$ denote matrices containing the actual and estimate values. Their (i,j)-th element are represented as $O_{i,j}$ and $\hat{O}_{i,j}$, respectively. M and D denote the number of data samples and the dimension of output variables.

$$\text{ARMSE} := \frac{1}{D}\sum_{j=1}^{D}\sqrt{\frac{\sum_{i=1}^{M}(\hat{O}_{i,j}-O_{i,j})^2}{M}}\,. \tag{2.20}$$

- Mean absolute percentage error (MAPE): The MAPE of the j-th response is:

$$\text{MAPE} := \frac{1}{M}\sum_{i=1}^{M}\left|\frac{\hat{O}_{i,j}-O_{i,j}}{O_{i,j}}\right| \times 100\%\,. \tag{2.21}$$

- Average Wasserstein distance (AWD): Wasserstein distance can measure the distance between two probability distributions [80]. Let ρ_j and $\hat{\rho}_j$ denote the probability distributions of the j-th column of $\mathbf{O}$ and $\hat{\mathbf{O}}$, respectively. The first-order Wasserstein distance loss between them can be calculated as:

$$l_{wd}(\hat{\rho}_j,\rho_j) = \inf_{\gamma\in\Gamma(\hat{\rho}_j,\rho_j)}\int_{\mathbb{R}\times\mathbb{R}}|\hat{\rho}_j-\rho_j|\,\mathrm{d}\gamma(\hat{\rho}_j,\rho_j)\,, \tag{2.22}$$

where $\Gamma(\hat{\rho}_j,\rho_j)$ represents the set of all measures on $\mathbb{R}\times\mathbb{R}$ whose marginal distributions are $\hat{\rho}_j$ and ρ_j on the first and second factors. Thus, the average Wasserstein distance of all responses can be obtained as:

$$\text{AWD} := \frac{1}{D}\sum_{j=1}^{D} l_{wd}(\hat{\rho}_j,\rho_j). \tag{2.23}$$

2.2.3.5 PF analysis results

Based on Table 2.2, we can draw several conclusions:

- The proposed NN structure with the designed initialization method is more accurate than the other competing methods. For example, the Data-driven, Linearized PF, and Jacobian methods are around 2.1, 2.7, and 2.8 times smaller than the

FC method on average. The proposed initialization schemes outperform random initialization.

- The performance of non-linear solvers is better than the linear solvers such as *Cumulants* and *RR* methods.

- The *ResNet* and *Random* methods do not outperform the FC method significantly, which means that the advantage of only introducing shortcut connection layers is not evident.

Table 2.2: ARMSEs of voltage phasor calculations in different cases (10^{-4})

Cases	Voltage phasor	*Cumulants*	*RR*	*FC*	*TPBNN*	*ResNet*	*Random*	*Data-driven*	*Linearized PF*	*Jacobian*
IEEE-30	angle	188.40	3.66	2.96	6.29	2.17	2.06	1.43	**1.35**	1.94
	magnitude	13.72	1.78	2.17	5.20	1.44	1.23	1.08	**1.02**	1.16
IEEE-118	angle	289.31	24.33	7.36	61.37	8.14	7.14	2.70	**2.46**	2.72
	magnitude	11.93	1.43	4.07	9.84	5.09	2.93	**0.79**	1.24	0.85
IEEE-300	angle	302.38	108.55	21.71	32.66	35.11	23.59	12.87	7.80	**6.96**
	magnitude	27.94	11.23	5.58	8.20	8.81	10.65	2.42	2.18	**1.94**
SouthCarolina-500	angle	309.28	259.15	16.18	798.32	13.40	7.83	8.95	**6.93**	8.26
	magnitude	58.46	46.68	10.91	15.13	8.48	3.35	3.67	**3.31**	3.43
PEGASE-1354	angle	207.19	164.72	32.20	-	30.84	13.79	12.77	9.13	**8.78**
	magnitude	7.82	6.59	10.23	-	10.11	15.53	4.34	**3.53**	3.54

Table 2.3: ARMSEs of the branch flow calculations for different cases

Cases	Branch flow	*Cumulants*	*RR*	*FC*	*TPBNN*	*ResNet*	*Random*	*Data-driven*	*Linearized PF*	*Jacobian*
IEEE-30	active	3.998	0.038	0.108	0.618	0.057	0.029	**0.028**	0.031	**0.028**
	reactive	1.064	0.048	0.103	0.457	0.056	0.036	0.034	0.041	**0.033**
IEEE-118	active	2.836	0.282	0.391	3.585	0.745	0.255	**0.071**	0.105	0.077
	reactive	1.341	0.153	0.321	1.654	0.440	0.221	0.067	0.108	**0.065**
IEEE-300	active	0.930	**0.209**	1.718	2.430	4.526	5.146	0.496	0.557	0.508
	reactive	2.314	0.686	0.924	2.080	1.817	2.869	0.471	0.452	**0.427**
SouthCarolina-500	active	1.658	1.427	4.394	17.896	2.834	0.970	0.756	0.767	**0.741**
	reactive	3.843	3.095	3.015	81.021	2.091	1.205	0.909	**0.798**	0.859
PEGASE-1354	active	**1.103**	1.131	16.164	-	15.843	11.94	7.066	5.808	5.922
	reactive	2.197	**1.857**	20.48	-	8.189	19.65	6.841	5.395	5.406

As shown in Fig. 2.2 and Fig. 2.3, the average MAPEs of phase angles and voltage magnitudes of the Data-driven method are 0.028% and 0.0042%, similarly, 0.023% and 0.0065% of the Linearized PF method, and 0.023% and 0.0042% of the Jacobian method.

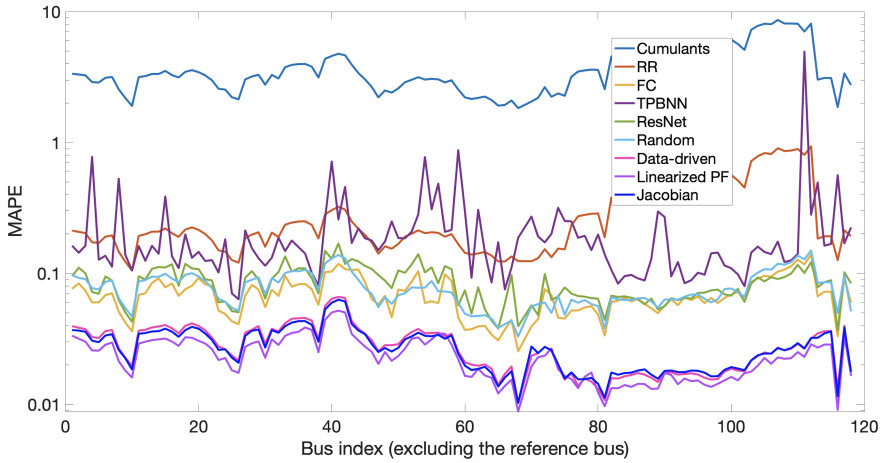


Figure 2.2: The MAPEs of phase angle calculations on the IEEE-118 bus system.

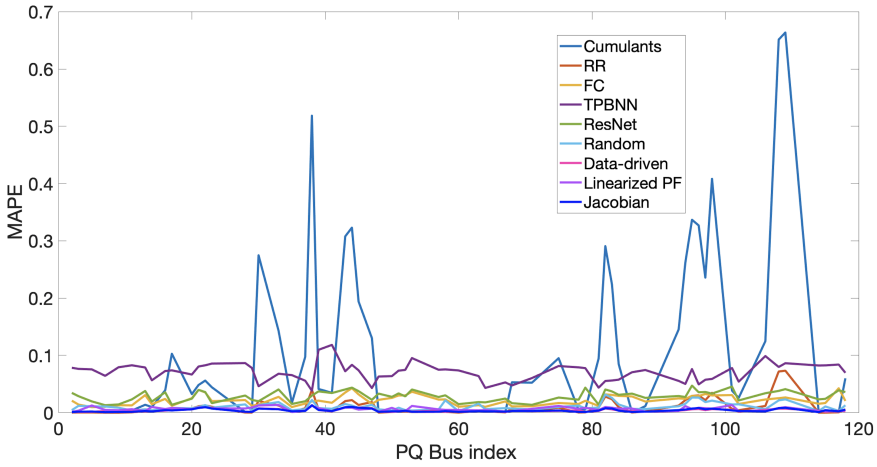


Figure 2.3: The MAPEs of voltage magnitude calculations on the IEEE-118 bus system.

Table 2.3 shows the ARMSEs of active and reactive branch flows. Despite inaccurate voltage phasor estimates, the *RR* and *Cumulants* methods achieve the best performance of the active branch flow estimates on the IEEE-300 and PEGASE-1354 bus systems, respectively. The reason is as follows. The output variables are voltage angles instead of voltage angle differences. The branch flow between two connected buses depends on their phase angle difference. Thus, smaller voltage angle estimate errors do

not necessarily imply smaller estimation errors of the voltage angle differences, which may lead to more significant branch flow estimate errors.

2.2.3.6 Wasserstein distance comparison results

Wasserstein distance calculates the minimum effort of transforming the probability mass from one distribution to another. It can serve as a good criterion to evaluate the probabilistic distribution difference between the estimate and the ground truth. Table 2.4 and Table 2.5 show the AWD of voltage phasor and branch flow distributions. The physics-guided initialization methods achieve better results than the *Random* and *Data-driven* methods.

Table 2.4: AWDs of voltage phasor distributions in different cases (10^{-4})

Cases	Voltage phasor	Cumulants	RR	FC	TPBNN	ResNet	Random	Data-driven	Linearized PF	Jacobian
IEEE-30	angle	152.99	2.07	0.86	2.03	0.70	0.76	0.53	**0.49**	0.70
	magnitude	10.73	0.59	0.51	1.99	0.35	0.37	0.28	**0.25**	0.32
IEEE-118	angle	257.96	12.7	1.96	10.43	2.22	2.48	0.92	**0.86**	0.91
	magnitude	10.17	0.58	0.94	2.98	1.32	0.74	0.22	0.52	**0.20**
IEEE-300	angle	266.14	69.72	4.57	5.81	7.07	4.75	3.49	2.36	**2.22**
	magnitude	22.44	4.08	1.29	2.53	1.78	3.14	0.60	0.62	**0.52**
SouthCarolina-500	angle	167.55	160.44	3.26	59.16	2.82	1.75	2.44	**1.71**	1.82
	magnitude	35.30	29.06	1.62	3.24	1.22	1.05	1.19	0.97	**0.89**
PEGASE-1354	angle	108.80	101.01	5.79	-	5.66	5.22	3.92	2.79	**2.72**
	magnitude	4.75	2.40	2.26	-	2.75	5.32	1.53	**1.04**	1.09

Table 2.5: AWDs of branch flow distributions in different cases

Cases	Branch flow	Cumulants	RR	FC	TPBNN	ResNet	Random	Data-driven	Linearized PF	Jacobian
IEEE-30	active	3.293	0.019	0.024	0.246	0.019	0.013	0.016	0.018	**0.012**
	reactive	0.871	0.014	0.025	0.194	0.014	0.012	0.014	0.025	**0.010**
IEEE-118	active	2.420	0.145	0.121	1.740	0.213	0.147	0.045	0.080	**0.039**
	reactive	1.144	0.074	0.117	0.715	0.160	0.115	0.039	0.085	**0.030**
IEEE-300	active	0.781	**0.134**	0.644	1.043	2.066	2.894	0.249	0.268	0.256
	reactive	1.880	0.325	0.396	1.049	0.907	2.073	0.347	**0.289**	0.296
SouthCarolina-500	active	0.924	0.860	1.273	6.772	0.922	0.540	0.395	**0.356**	**0.356**
	reactive	2.278	1.935	1.176	7.537	0.803	0.922	0.681	0.569	0.630
PEGASE-1354	active	**0.611**	0.629	7.089	-	6.815	9.806	4.493	3.077	3.148
	reactive	1.274	**0.890**	4.523	-	5.277	14.079	5.105	3.497	3.589

2.2.3.7 PDF estimation comparison results

As shown in Fig. 2.4 and Fig. 2.5, the proposed methods achieve smaller estimation errors in the PDF estimates. It is worth pointing out that the voltage magnitude of bus 16 has the most significant std value. Hence, we plot its PDF to verify that the proposed methods can track the voltage magnitude with large fluctuations. As shown in Fig. 2.5, the *RR* method cannot accurately estimate the highly skewed PDF.

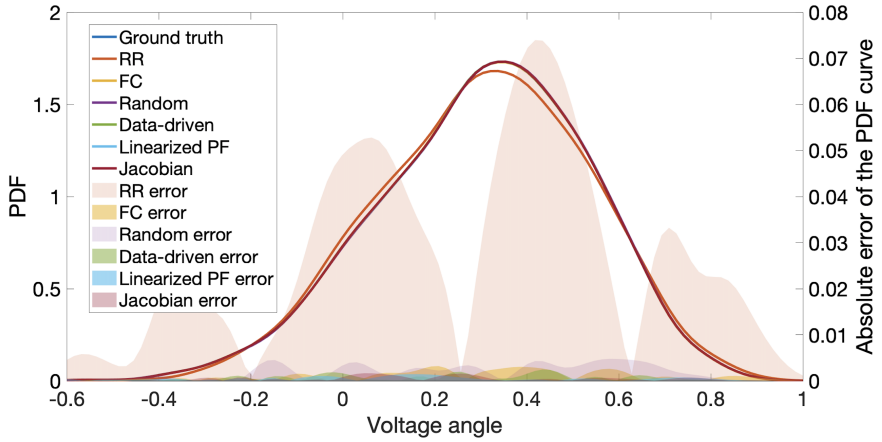


Figure 2.4: Voltage angle of bus 1 for the IEEE-300 bus system.

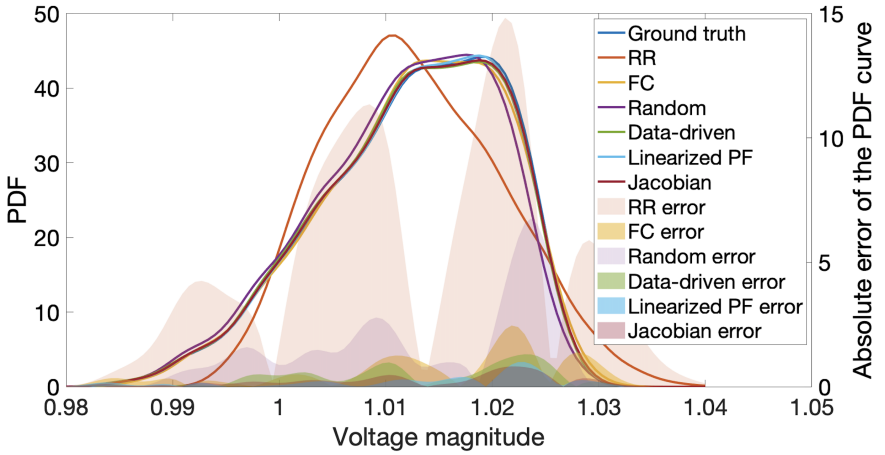


Figure 2.5: Voltage magnitude of bus 16 for the IEEE-300 bus system.

2.2.3.8 Computational time

The testing time is short because the forward propagation is fast. Table 2.6 shows the training time of different NN structures. Compared to MLPs, residual blocks need extra time due to the backward propagation of the shortcut connection.

Table 2.6: The training time of each epoch for different NN structures (seconds).

Cases	FC	TPBNN	ResNet	Proposed approaches
30	0.87	2.32	1.13	1.07
118	1.55	4.11	1.90	1.91
300	2.13	6.27	1.97	2.37
500	3.57	15.66	4.57	3.74
1354	4.58	-	6.10	5.77

2.2.3.9 Convergence rate

For a fair comparison, we use the same learning rate 10^{-4} for all the NN-based methods to demonstrate the evolution process of the training loss. As shown in Fig. 2.6 and Fig. 2.7, after training the first epoch, the proposed initialization schemes have achieved much smaller error values than the other methods. A faster convergence rate is appealing whenever the training time is limited.

In addition, we observe that the designed three initialization schemes converge faster than random initialization. Thus, we demonstrate how initial weights will affect the entire training process. Fig. 2.8a shows the initial weights of the shortcut connection layer under random initialization. After training 500 epochs, the pattern of the NN weights still looks random, as shown in Fig. 2.8b. In contrast, using the Data-driven initialization method, Fig. 2.9 shows that the pattern of updated weights is quite similar

to that of the initial weights. This phenomenon indicates that initialization methods affect the NN updates during training.

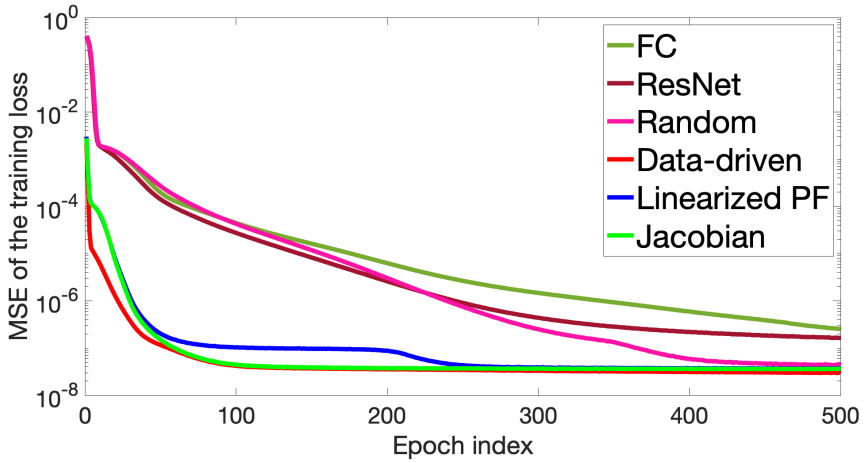


Figure 2.6: The training loss evolution process (starting from when the first epoch's training is done) on the IEEE-30 bus system.

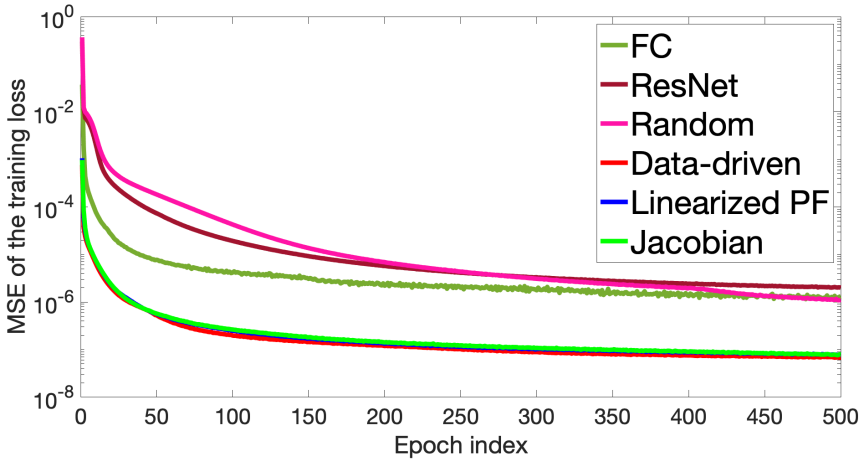
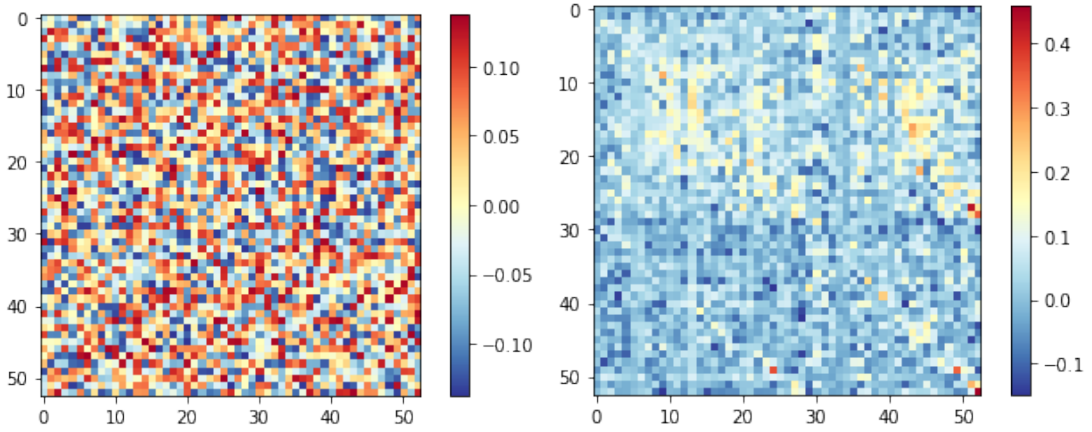


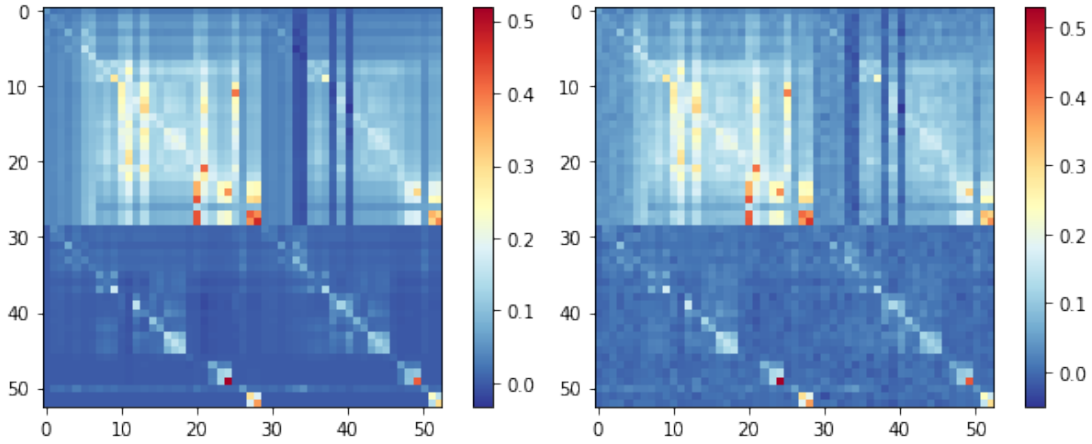
Figure 2.7: The training loss evolution process (starting from when the first epoch's training is done) on the IEEE-118 bus system.



(a) Initial weights

(b) Updated weights after 500 epochs' training

Figure 2.8: Weights of the shortcut connection linear layer of the *Random* method for the IEEE-30 bus system.



(a) Initial weights

(b) Updated weights after 500 epochs' training

Figure 2.9: Weights of the shortcut connection linear layer of the *Data-driven* method for the IEEE-30 bus system.

2.3 Variation-cognizant PPF via Multi-task Learning

Based on the branch flow equations, the branch flow between two connected buses is related to the voltage angle difference compared to the voltage angles. The training loss function used in section 2.2 consists of the voltage magnitude and phase angle estimation errors. Hence, the proposed NN framework shows more evident advantages in the voltage phase estimates than the branch flow estimates. In this section, we propose two separate learning frameworks for the voltage magnitude and phase angles to improve branch flow estimation accuracy. First, we combine linear regression and the FCNN to predict the voltage magnitudes. Then, we employ multi-task learning to enhance the accuracy of branch flow calculations by incorporating the voltage angle difference errors into the loss function design.

2.3.1 Proposed methods for PF Analysis

Based on the AC-PF equations (2.1), the mapping from the known power injections to the unknown phase angles and voltage magnitudes can be compactly rewritten as:

$$\mathbf{y}_a = \mathbf{f}_a(\mathbf{x})\,, \tag{2.24}$$

$$\mathbf{V}_l = \mathbf{f}_m(\mathbf{x})\,, \tag{2.25}$$

where $\mathbf{y}_a = [\boldsymbol{\theta}_l; \boldsymbol{\theta}_g]$ collect the unknown phase angles. We adopt two different learning frameworks to approximate the mapping from $\mathbf{x}$ to the voltage magnitudes $\mathbf{V}_l$ and

voltage angles $\mathbf{y}_a$, respectively.

2.3.1.1 Hybrid learning framework for voltage magnitudes

In a per-unit power system, voltage magnitudes are typically around 1; thus, the voltage magnitude fluctuations are small. The std values of voltage magnitudes can be obtained based on the historical operational data. We split the load buses into two disjoint subsets according to the std values:

$$\mathcal{N}_l = \mathcal{N}_{ls} \cup \mathcal{N}_{lb} \quad \text{and} \quad \mathcal{N}_{ls} \cap \mathcal{N}_{lb} = \emptyset, \tag{2.26}$$

where $\mathcal{N}_{ls} := \{\mathcal{N}_l \mid \text{std}(V_l) \leq \gamma\}$ collects the load buses with small std values while $\mathcal{N}_{lb} := \{\mathcal{N}_l \mid \text{std}(V_l) > \gamma\}$ collects the load buses with large std values. The threshold parameter γ can be learned via the validation process.

We propose two different learning strategies for these two subsets. The NN has shown an impressive capability in non-linear function approximation. Hence, for the load buses in $\mathcal{N}_{ld}$, we employ the NN to learn the inverse AC-PF mapping from the power injections to voltage magnitudes. However, if the voltage magnitude remains nearly constant regardless of the variations of power injections, the complicated NN may lead to overfitting. Hence, we adopt ordinary least-squares linear regression for the load buses in $\mathcal{N}_{lc}$.

2.3.1.2 Branch flows computation via Multi-task Learning

The motivation for introducing multi-task learning for phase angle estimation is as follows. As shown in (2.2), branch flows depend on the voltage angle differences between the connected buses instead of voltage angles. In the meantime, the relationship between phase angle and phase angle difference is not bijective. The phase angles uniquely determine the phase angle differences, while the opposite does not hold. Thus, accurate phase angle estimates do not always ensure accurate phase angle difference estimates. In the first work, the loss function only includes the errors of voltage angle estimates. Therefore, we consider incorporating the estimation errors of voltage angle differences to help improve branch flow estimation accuracy. Hence, the joint training loss function is designed as follows:

$$\mathcal{L}_{\text{new}} := \mathcal{L}(\Delta \mathbf{y}_{\text{a}}) + \alpha_a \mathcal{L}(\Delta \mathbf{y}_{\text{ad}}) := \mathcal{L}(\bar{\mathbf{y}}_{\text{a}} - \mathbf{y}_{\text{a}}) + \alpha_a \mathcal{L}(\bar{\mathbf{y}}_{\text{ad}} - \mathbf{y}_{\text{ad}}), \tag{2.27}$$

where $\mathcal{L}(\cdot)$ is the mean square error, and α_a is used to balance the relative importance of two tasks. Let $\mathbf{A} \in \mathbb{R}^{M \times (N-1)}$ denote the reduced incidence adjacency matrix (the column related to the slack bus is deleted) whose element is 1 for the *from node* while -1 for the *end node* of a branch. Voltage angle differences can be calculated by $\mathbf{y}_{\text{ad}} = \mathbf{A}\mathbf{y}_a$, and the voltage angle difference estimates $\bar{\mathbf{y}}_{\text{ad}}$ can be obtained in a similar fashion.

2.3.2 Simulation Results

This section verifies the effectiveness of the proposed methods in solving voltage phasors and branch flows and estimating their probabilistic properties on the IEEE-300

and PEGASE-1354 bus systems.

2.3.2.1 Data generation

The solar panel outputs and active power demands are generated based on the data in [75] and [74], respectively. The actual data samples are insufficient for our simulation; thus, we use multivariate Gaussian distribution as a synthetic data supplement. The mean value refers to the nominal value, and the ratio of mean value to std is 0.01 and 0.1 on the IEEE-300 and PEGASE-1354 bus systems, respectively. The correlation coefficient of active and reactive power injections for the same bus is 0.8 and 0.2 for different buses.

2.3.2.2 Methods for comparison

We compare the proposed methods (M3 and M4) with two existing approaches (M1 and M2):

- M1: Assume a linear relationship $\begin{bmatrix} \mathbf{y}_a \\ \mathbf{V}_l \end{bmatrix} = \mathbf{H}\mathbf{x} + \boldsymbol{\epsilon}$, and use least-squares linear regression to predict the voltage magnitudes and phase angles.
- M2 (cf. [41]): Train one FCNN to predict both voltage magnitudes and phase angles simultaneously.
- M3: Train two separate FCNNs to predict the voltage magnitudes and phase angles, respectively.

- M4: Use the hybrid learning method and multi-task learning method to estimate the voltage magnitudes and phase angles, respectively.

2.3.2.3 Training details

The mini-batch training with Adam optimizer on `Pytorch 1.7.1` in `Python 3.7` is adopted [78]. The mini-batch size is 32, and the activation function is ReLu. α_a is set to 1 and 10 on the IEEE-300 and PEGASE-1354 bus systems, respectively. Table 2.7 shows the data size, NN structure, and learning rate. We train and test these methods five times to alleviate the randomness.

Table 2.7: Hyperparameters of different methods on different bus systems

Bus systems	[Training, Validation, Testing]	Methods	Voltage phasors	Neural network structure	Learning rate
IEEE-300	[20k, 5k, 5k]	M2	angle and magnitude	[530 200 200 200 530]	1×10^{-4}
		M3	angle	[530 300 300 299]	5×10^{-5}
			magnitude	[530 200 200 231]	1×10^{-4}
		M4	angle	[530 300 300 299]	5×10^{-5}
			magnitude	[530 200 200 137]	1×10^{-4}
PEGASE-1354	[26k, 6k, 6k]	M2	angle and magnitude	[2447 300 300 300 300 2447]	7×10^{-5}
		M3	angle	[2447 400 400 400 1353]	5×10^{-5}
			magnitude	[2447 400 400 400 1094]	7×10^{-5}
		M4	angle	[2447 400 400 400 1353]	7×10^{-5}
			magnitude	[2447 400 400 400 718]	7×10^{-5}

2.3.2.4 Evaluation criteria

Three evaluation criteria are employed to provide a comprehensive evaluation of different methods.

- Average root mean square error (RMSE) of all responses:

$$\text{Average RMSE} := \frac{1}{d}\sum_{i=1}^{d}\sqrt{\frac{1}{n}\|\bar{\mathbf{O}}(:,i) - \mathbf{O}(:,i)\|_2^2}\,, \tag{2.28}$$

where $\mathbf{O}(:, i)$ represents the i-th response (i.e., column of matrix $\mathbf{O}$), and similarly for $\bar{\mathbf{O}}(:, i)$.

- Average Wasserstein distance [80]:

$$\text{AWD} := \frac{1}{d}\sum_{i=1}^{d} \mathcal{W}_1(\bar{\rho}_i\,, \rho_i)\,, \tag{2.29}$$

where $\mathcal{W}_1$ refers to the first-order Wasserstein distance, and $\bar{\rho}_i$ and ρ_i denote the estimates and targets of probability distributions of the i-th response, respectively.

- Average mean absolute error (MAE) of the mean value e_1 and the std value e_2:

$$e_1 := \frac{1}{d}\sum_{i=1}^{d} |\bar{\mu}_i - \mu_i|, \quad e_2 := \frac{1}{d}\sum_{i=1}^{d} |\bar{\sigma}_i - \sigma_i|. \tag{2.30}$$

where $\bar{\mu}_i$ and μ_i denote the mean values of the i-th response, and $\bar{\sigma}_i$ and σ_i are their corresponding std values.

2.3.2.5 PF analysis results

Table 2.8 shows that the proposed hybrid learning method performs best in the voltage magnitude estimates. As shown in Tables 2.9 and 2.10, the proposed multi-task learning method improves the estimation accuracy of the voltage angle differences, yielding more accurate branch flow estimates. M4 performs slightly worse than M3 in the voltage angle estimates on the PEGASE-1354 bus system. The reason is that it is hard to ensure all tasks reach their optimalities simultaneously in multi-task learning. Since

these two tasks are highly related, minimizing the voltage angle difference estimation error does not significantly degrade the voltage angle estimate accuracy.

Table 2.8: Average RMSEs of voltage magnitudes (10^{-4})

Cases	M1	M2	M3	M4
IEEE-300	16.15	4.36	3.68	**3.34**
PEGASE-1354	2.58	6.81	2.75	**2.11**

Table 2.9: Average RMSEs of voltage angles and angles differences calculations (10^{-3})

Cases	Voltage	M1	M2	M3	M4
IEEE-300	angle	23.15	2.31	2.24	**2.10**
	angle difference	0.91	0.51	0.50	**0.38**
PEGASE-1354	angle	5.96	1.53	**0.97**	1.11
	angle difference	0.84	0.81	0.59	**0.53**

Table 2.10: Average RMSEs of branch flows calculations

Cases	Branch flow	M1	M2	M3	M4
IEEE-300	active	2.92	1.51	1.69	**0.93**
	reactive	1.23	0.96	0.70	**0.56**
PEGASE-1354	active	22.12	8.67	7.89	**5.60**
	reactive	5.51	5.55	3.01	**2.03**

2.3.2.6 Probabilistic descriptions comparison results

Fig. 2.10 and Table 2.11 show the proposed hybrid learning method has the least AWD of voltage magnitude estimates. Besides, Fig. 2.11 and Table 2.12 illustrate that the proposed method M4 outperforms the other methods in probabilistic distributions of branch flows. As shown in Table 2.13, the proposed method M4 achieves the smallest MAEs in the reactive branch flow estimates. Besides, M4 reduces the MAE of the std values at least by half on the active branch flow estimates.

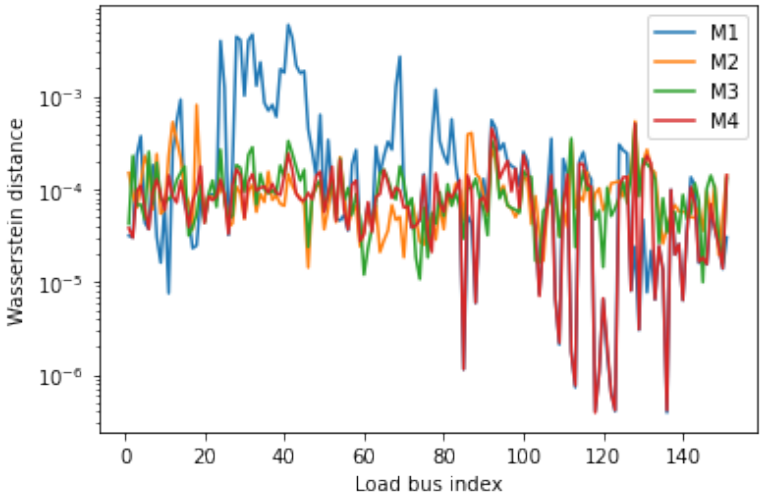
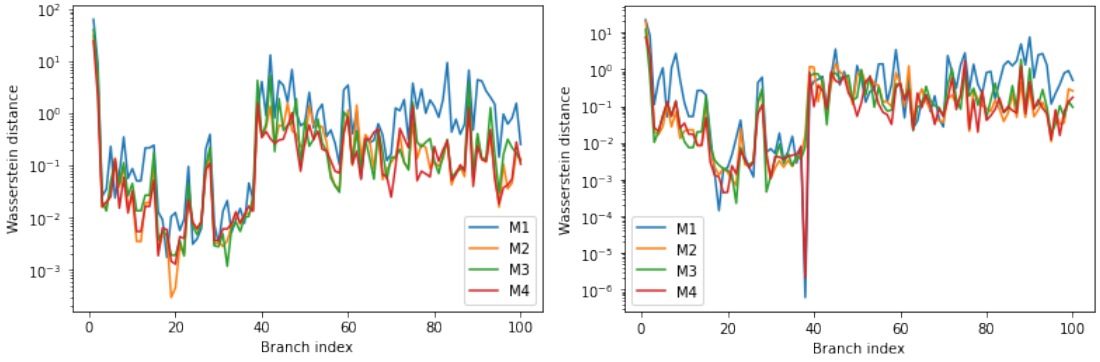


Figure 2.10: The Wasserstein distance of voltage magnitude distributions for the IEEE-300 bus system.



(a) Active branch flow (b) Reactive branch flow

Figure 2.11: The Wasserstein distance of branch flows distributions for the IEEE-300 bus system.

Table 2.11: AWD of the voltage phasors distributions (10^{-4})

Cases	Voltage	M1	M2	M3	M4
IEEE-300	angle	139.544	5.580	5.104	**5.102**
	magnitude	8.136	1.057	1.182	**0.917**
PEGASE-1354	angle	25.199	3.840	**3.066**	3.629
	magnitude	0.969	1.548	0.946	**0.674**

Table 2.12: AWD of the branch flows distributions

Cases	Branch flow	M1	M2	M3	M4
IEEE-300	active	1.843	0.527	0.772	**0.403**
	reactive	0.579	0.369	0.323	**0.228**
PEGASE-1354	active	16.511	3.975	3.853	**2.788**
	reactive	3.979	2.998	1.827	**1.177**

Table 2.13: Average MAE of estimate mean and std of branch flows

Cases	Branch flow		M1	M2	M3	M4
IEEE-300	active	mean	1.27	**0.16**	0.38	0.29
		std	1.27	0.57	0.72	**0.28**
	reactive	mean	0.22	0.19	0.23	**0.14**
		std	0.36	0.34	0.19	**0.11**
PEGASE-1354	active	mean	12.43	**1.79**	**1.79**	1.95
		std	12.98	4.30	4.13	**2.09**
	reactive	mean	2.88	1.75	1.32	**0.94**
		std	3.11	3.30	1.37	**0.70**

2.4 Summary

This chapter introduces a novel physics-informed NN learning framework to speed up the PF analysis. Three different initialization schemes are designed to utilize the linear property of AC-PF equations. The Linearized PF and Jacobian initialization schemes require accurate information on network topology and line parameters, while the Data-driven initialization only needs historical data. Tested on IEEE benchmark systems, extensive simulation results show that the proposed learning framework outperforms the competing methods in estimation accuracy and training efficiency.

Next, we propose a hybrid learning strategy for voltage magnitude estimates and multi-task learning to improve branch flow estimate accuracy. According to the std values of voltage magnitudes, two different data-driven models are employed to estimate

the voltage magnitudes. In addition, the joint loss, consisting of the voltage angles and voltage angle differences, is adopted to improve the accuracy of branch flow estimates. The simulation results show that the proposed methods achieve promising performance in estimating branch flows and their probability descriptions.

Chapter 3

Optimal Power Flow Analysis

This section introduces the AC-OPF problem formulation and depicts two proposed end-to-end learning-based frameworks for the AC-OPF analysis. We propose an unsupervised learning-based framework that utilizes the Lagrangian dual function as the training loss function. Compared to the fixed weighting parameter in the training loss function, we adopt dynamically updated Lagrange multipliers based on the constraint violation to guide the NN training process and improve the feasibility of the decision variables. However, the training time is long due to the complicated gradient computation, which prevents a daily NN update for accommodating new data instances. Hence, we develop novel batch-mean gradient estimation approaches based on the semi-supervised learning framework to reduce the data preparation and training time. The proposed gradient estimation method achieves a comparable convergence rate as the group truth gradient while significantly improving the training efficiency.

3.1 AC-OPF Problem Formulation

The AC-OPF problem minimizes the total generation cost while satisfying a set of operational constraints [81]:

$$\min_{\mathbf{V},\boldsymbol{\theta},\mathbf{P}_g,\mathbf{Q}_g} \quad \sum_i c_i(P_{g,i}) \tag{3.1a}$$

$$\text{s.t.} \quad P_{g,i} - P_{d,i} = V_i \sum_{j=1}^{N} V_j (G_{ij}\cos\theta_{ij} + B_{ij}\sin\theta_{ij}) \tag{3.1b}$$

$$Q_{g,i} - Q_{d,i} = V_i \sum_{j=1}^{N} V_j (G_{ij}\sin\theta_{ij} - B_{ij}\cos\theta_{ij}) \tag{3.1c}$$

$$P_{ij} = -G_{ij}V_i^2 + V_iV_j(G_{ij}\cos\theta_{ij} + B_{ij}\sin\theta_{ij}) \tag{3.1d}$$

$$Q_{ij} = B_{ij}V_i^2 + V_iV_j(G_{ij}\sin\theta_{ij} - B_{ij}\cos\theta_{ij}) \tag{3.1e}$$

$$P_{ij}^2 + Q_{ij}^2 = |S_{ij}|^2 \,, \forall (i,j) \in \mathcal{M} \tag{3.1f}$$

$$|S_{ij}|^2 \le (S_{ij}^{\max})^2, \; \forall (i,j) \in \mathcal{M} \tag{3.1g}$$

$$P_{g,i}^{\min} \le P_{g,i} \le P_{g,i}^{\max} \,, \forall i \in \mathcal{N} \setminus \mathcal{N}_d \tag{3.1h}$$

$$Q_{g,i}^{\min} \le Q_{g,i} \le Q_{g,i}^{\max} \,, \forall i \in \mathcal{N} \setminus \mathcal{N}_d \tag{3.1i}$$

$$V_i^{\min} \le V_i \le V_i^{\max}, \; \forall i \in \mathcal{N} \tag{3.1j}$$

$$\theta_{\text{ref}} = 0 \tag{3.1k}$$

$$P_{g,i} = Q_{g,i} = 0 \,, \forall i \in \mathcal{N}_d. \tag{3.1l}$$

$P_{g,i}$, $Q_{g,i}$, $P_{d,i}$ and $Q_{d,i}$ denote the active and reactive power generations and load demands at bus i. V_i denotes the voltage magnitude of bus i. $\theta_{ij} := \theta_i - \theta_j$ represents the voltage angle difference between bus i and j. P_{ij} and Q_{ij} denote the active and

reactive branch flows from bus i to bus j. G_{ij} and B_{ij} are the real and imaginary parts of the (i,j)-th element of nodal admittance matrix $\mathbf{Y} \in \mathbb{C}^{N\times N}$, respectively. In addition, Eq.(3.1a) is the objective function that captures the total active power generation costs, where $c_i(\cdot)$ is the power generation cost function of generator i. Eqs.(3.1b) and (3.1c) refer to the nodal power balance equations based on Kirchhoff's law. Eqs.(3.1d) and (3.1e) represent the branch flow balance equations, and Eq.(3.1g) gives the upper limits of apparent power flows. Eqs.(3.1h)-(3.1i) depict the operational constraints for the active and reactive power generation outputs. Eq.(3.1j) depicts the operating limits of voltage magnitudes. The phase angle of the slack bus is set to 0. Eq.(3.1l) implies that the load buses do not connect to any generators.

3.2 Unsupervised Learning Framework for AC-OPF Analysis

This section depicts the proposed NN learning framework and describes the training loss function based on the Lagrangian dual function.

3.2.1 AC-OPF problem via Lagrangian Function

The AC-OPF problem (3.1) can be formulated as a generic optimization problem with inequality constraints, which can be solved by the augmented Lagrangian

method:

$$\min \quad f(\mathbf{y}, \mathbf{z}_2) \tag{3.2a}$$

$$\text{s.t.} \quad \mathbf{h}(\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2) \leq \mathbf{0}, \tag{3.2b}$$

where $f(\mathbf{y}, \mathbf{z}_2)$ refers to the objective function (3.1a), and the vector-valued function $\mathbf{h}(\cdot)$ collects all inequality constraints (3.1g)-(3.1j).

Lagrangian relaxation penalizes the inequality constraint violation values by introducing the Lagrange multipliers to the loss function. After relaxing all inequality constraints by adding the constraint violation penalties to the objective function, the augmented Lagrangian function can be expressed as [82]:

$$L(\mathbf{x}, \mathbf{y}, \mathbf{z}_1, \mathbf{z}_2, \boldsymbol{\mu}) = f(\mathbf{y}, \mathbf{z}_2) + \frac{1}{2\alpha_h} \mathbf{1}_d^\top * \left((\text{ReLu}(\boldsymbol{\mu} + \alpha_h \mathbf{h}(\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2)))^2 - \boldsymbol{\mu} \odot \boldsymbol{\mu} \right), \tag{3.3}$$

where $\boldsymbol{\mu} \in \mathbb{R}^d, d = M + 4(N_g + 1) + 2N$ collects the Lagrange multipliers associated with the inequality constraints. α_h is a constant coefficient. $\text{ReLu}(\cdot)$ denotes the element-wise rectified linear unit; $\odot$ denotes the element-wise product;

The dual function of Eq.(3.3) is given as:

$$g(\boldsymbol{\mu}) = \min_{\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2} L(\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2, \boldsymbol{\mu}). \tag{3.4}$$

The dual problem maximizes the dual objective (3.4) to provide the tightest lower bound

of $f(\mathbf{y}, \mathbf{z}_2)$:

$$\max_{\boldsymbol{\mu} \geq \mathbf{0}} \quad g(\boldsymbol{\mu}). \tag{3.5}$$

The primal-dual approach updates the primal and dual variables sequentially at each iteration to solve the dual problem. Let $\boldsymbol{\mu}_k$ denote the Lagrange multiplier vector at the k-th iteration. The primal update attempts to obtain the primal variables $\{\mathbf{y}_k, \mathbf{z}_{1,k}, \mathbf{z}_{2,k}\}$ by solving the dual function (3.4). Then, the Lagrange multiplier at the next iteration can be calculated by:

$$\boldsymbol{\mu}_{k+1} := \mathrm{ReLu}(\boldsymbol{\mu}_k + \alpha_h \mathbf{h}(\mathbf{y}_k, \mathbf{z}_{1,k}, \mathbf{z}_{2,k})). \tag{3.6}$$

3.2.2 NN Learning framework for AC-OPF Analysis

Let $(\mathbf{P}_g)_{\mathcal{N}_g}$ and $(\mathbf{V})_{\mathcal{N}_g}$ denote the active power generations and voltage magnitudes of generator buses, respectively. $P_{g,\mathrm{ref}}$ and $Q_{g,\mathrm{ref}}$ denote the active and reactive power generations of the slack bus, respectively. Let $\mathbf{x} = [(\mathbf{P}_d)_{\mathcal{N}}; (\mathbf{Q}_d)_{\mathcal{N}}] \in \mathbb{R}^{2N}$ collect the load demands of all buses. Let $\mathbf{y} = [(\mathbf{P}_g)_{\mathcal{N}_g}; (\mathbf{V})_{\mathcal{N}_g}; V_{\mathrm{ref}}; \theta_{\mathrm{ref}}] \in \mathbb{R}^{2N_g+2}$ collect a partial set of decision variables. The remaining decision variables are collected in $\mathbf{z}_1 = [(\mathbf{V})_{\mathcal{N}_d}; \boldsymbol{\theta}_{\mathcal{N}_g \cup \mathcal{N}_d}] \in \mathbb{R}^{2N_d+N_g}$ and $\mathbf{z}_2 = [P_{g,\mathrm{ref}}; Q_{g,\mathrm{ref}}; (\mathbf{Q}_g)_{\mathcal{N}_g}, \mathbf{S}_{ij}^2] \in \mathbb{R}^{N_g+M+2}$. Let $\mathbf{v} = [\boldsymbol{\theta}; \mathbf{V}] \in \mathbb{R}^{2N}$ collect the phase angles and voltage magnitudes of all buses. Fig. 3.1 shows the schematic of the proposed learning framework. The FCNN is utilized to approximate the mapping $\mathbf{x} \mapsto \mathbf{y}$. Given $\mathbf{x}$ and $\mathbf{y}$, the PF solver can solve the PF

equations (3.1b) and (3.1c) to obtain the phase angles and voltage magnitudes. Given the voltage phasors, $\mathbf{z}_2$ can be determined based on the power flow and branch flow equations. Let $\mathbf{f}_r(\cdot)$ represent the mapping $\mathbf{v} \mapsto \mathbf{z}_2$, which can be obtained by solving the equality constraints $(3.1b)_{\text{ref}}$, $(3.1c)_{\text{ref}}$, $(3.1b)_{\mathcal{N}_g}$, and (3.1d)-(3.1f).

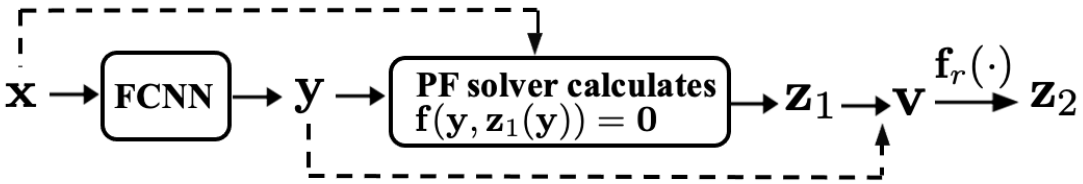


Figure 3.1: The proposed unsupervised learning framework for solving AC-OPF.

Remark. *The motivation for the decision variable splitting scheme is as follows [19]. In PF analysis, when* $\mathbf{x}$ *and* $\mathbf{y}$ *are available, we can build* $2N_d+N_g$ *power balance equations, which is a subset of Eqs.*(3.1b)-(3.1c)*. The unknown voltage magnitudes and phase angles in* $\mathbf{z}_1$ *can be obtained by solving the PF equations via the PF solver. Once the voltage magnitudes and phase angles of all buses are known,* $\mathbf{z}_2$ *are uniquely determined by the remaining equality constraints. Therefore, the variable splitting scheme ensures the satisfaction of branch flow and power balance.*

Let $\mathbf{y} := \mathcal{W}(\mathbf{x})$, where $\mathcal{W}$ collects the NN weights. Assume the FCNN only has one hidden layer. The mapping from the input $\mathbf{x}$ to the output $\mathbf{y}$ can be calculated as:

$$\mathbf{y} = \mathcal{B}(\text{Sigmoid}(\mathbf{W}_2\text{ReLu}(\mathbf{W}_1\mathbf{x}))) \,, \tag{3.7}$$

where $\mathbf{W}_i$ denotes the weight matrix of i-th linear layer. Sigmoid$(\cdot)$ is the activation function of the output layer. $\mathcal{B}(\cdot)$ is a linear operator to ensure that the output variables satisfy their box constraints. For example, let $\beta_i \in [0, 1]$ denote the NN output for voltage magnitude at bus i. Then, the decision variable $V_i \in [V_i^{\min}, V_i^{\max}]$ can be recovered by:

$$V_i = \mathcal{B}(\beta) := \beta_i V_i^{\min} + (1 - \beta_i) V_i^{\max}. \tag{3.8}$$

We implement similar transformations to all decision variables in $\mathbf{y}$ to satisfy the related box constraints.

3.2.3 Combine NN training with Lagrangian Duality

As shown in Fig. 3.1, $\mathbf{y}$ is computed by the forward propagation, i.e., $\mathbf{y} := \mathcal{W}(\mathbf{x})$. After plugging $\mathbf{y} = \mathcal{W}(\mathbf{x})$ and $\{\mathbf{z}_1, \mathbf{z}_2\} = \mathbf{u}(\mathcal{W}(\mathbf{x}))$ into the augmented Lagrangian function (3.3), we can express the Lagrangian function parameterized by the NN weights $\mathcal{W}$ as:

$$\mathcal{L}_{\mathcal{W}} := L(\mathcal{W}(\mathbf{x}), \mathbf{u}(\mathcal{W}(\mathbf{x})), \boldsymbol{\mu}). \tag{3.9}$$

We adopt the Lagrangian function $\mathcal{L}_{\mathcal{W}}$ as the training loss function. Algorithm 1 shows the proposed learning framework. The trained FCNN can serve as a rapid AC-OPF solver. For any input $\mathbf{x}$, the FCNN can rapidly predict $\mathbf{y}$. The other decision variables in $\mathbf{z}_1$ and $\mathbf{z}_2$ can be obtained based on the power flow and branch flow equations.

Remark. *The proposed learning framework does not rely on the training data pairs for*

Algorithm 1 Unsupervised learning framework via Lagrangian function

Input: Training dataset $\mathcal{X}$, coefficient α_h, initial value of multiplier $\boldsymbol{\mu}_0$, maximum training epoch n, multiplier updating period m.

for epoch $i = 1, 2, \ldots, n$ **do**:
  Sample data points $\mathbf{x} \in \mathcal{X}$.
  Compute $\mathbf{y}$ through feedforward propagation.
  Obtain $\mathbf{z}_1$ usingthe PF solver.
  Compute $\mathbf{z}_2$ according to Eqs. (3.1b)-(3.1f).
  Calculate the loss function (3.9), and update $\mathcal{W}$ via backpropagation.
  $\boldsymbol{\mu}_{i+1} \leftarrow \boldsymbol{\mu}_i$.
  if $i \mod m \equiv 0$ **then**
    $\boldsymbol{\mu}_{i+1} \leftarrow \text{ReLu}(\boldsymbol{\mu}_i + \alpha_h \mathbf{h}(\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2))$.
  end if
end for

NN updates. Therefore, it falls into the category of unsupervised learning. According to the constraint violation degree in Eq.(3.6), the penalty term in the loss function is dynamically adjusted via the periodic update of the Lagrange multiplier $\boldsymbol{\mu}$. The proposed approach can better guide the NN training process than the fixed penalty coefficient, which improves the solution feasibility.

3.2.4 Simulation Results

This section shows the performance of our proposed approach based on the IEEE-30 and IEEE-118 bus systems.

3.2.4.1 Data generation

The nominal values of load demands are denoted as $(\tilde{\mathbf{P}}_d)_{\mathcal{N}}$ and $(\tilde{\mathbf{Q}}_d)_{\mathcal{N}}$. We generate 5000 samples that are uniformly distributed over $[0.9\tilde{\mathbf{P}}_d, 1.1\tilde{\mathbf{P}}_d]$ and $[0.9\tilde{\mathbf{Q}}_d, 1.1\tilde{\mathbf{Q}}_d]$, with a training/validation/testing ratio of 10:1:1.

3.2.4.2 Methods for comparison

The proposed learning framework that uses the NR and FDPF solvers are named *NR-Dual* and *FDPF-Dual*, respectively. We compare them with the conventional solver *MIPS* and two existing unsupervised learning-based methods *DC3* and *NGT*. The proposed methods and *DC3* adopt the variable splitting scheme VS_1 to guarantee satisfying power balance equations. The *NGT* method uses the variable splitting scheme VS_2, which may lead to load mismatches at load buses.

- VS_1 (cf. Fig 3.1): The NN output is $\mathbf{y}$. Given $\mathbf{x}$ and $\mathbf{y}$, the decision variables in **z1** can be obtained by solving the AC-PF equations via the follow-up PF solver. Given $\mathbf{v}$, the decision variables in $\mathbf{z}_2$ can be obtained through $\mathbf{z}_2 = \mathbf{f}_r(\mathbf{v})$.

- VS_2 (cf. Fig 3.2): The NN output is $\mathbf{v}$, and the remaining decision variables in $\mathbf{P}_g$ and $\mathbf{z}_2$ are obtained by the power flow and branch flow equations, i.e., Eqs.(3.1b)-(3.1e)

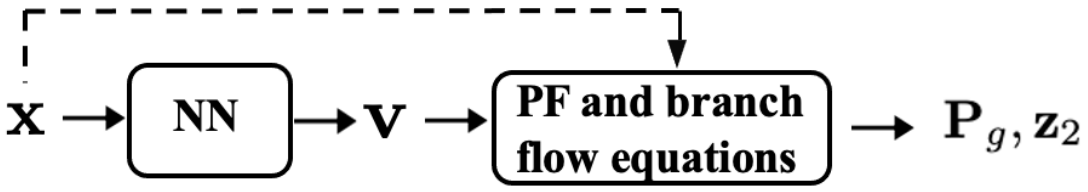


Figure 3.2: The proposed unsupervised learning framework using VS_2 for solving AC-OPF.

3.2.4.3 Training details

We implement the simulations on a server with NVIDIA Titan RTX GPU with 25GB of RAM. The Adam optimizer is used for NN training based on `Pytorch 1.7.1`. The batch size is 32, and the number of the training epoch is $n = 1000$. The FCNN has one hidden layer, and the number of neurons of the hidden layer is 50 and 100 for the IEEE-30 and IEEE-118 bus systems, respectively. The PF solver stops iterations when the norm of load mismatches is less than 10^{-5}. We update the Lagrange multiplier every 10 epoch instead of every epoch to help stabilize the training process [83]. Finally, α_h is set to 2.

3.2.4.4 Evaluation criteria

We have the following performance evaluation metrics for a comprehensive evaluation:

1. **Optimality**: The total generation cost.

2. **Feasibility**: The feasibility rate is obtained using the ratio of the number of satisfied inequality constraints to the total number of inequality constraints. In addition, we calculate the mean and maximum values of $\boldsymbol{\nu} := \text{ReLu}(\mathbf{h}(\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2))$ to evaluate the constraint violation degree.

3. **Load mismatch**: The relative error of the reconstructed load demands and the input load demands.

4. **Computational efficiency**: The computational time.

3.2.4.5 Feasibility of the proposed method

Table 3.1 shows the average nominal values of the decision variables in the per unit system (base value is 100 MVA). The nominal value can be a reference for evaluating the constraint violation degree. Table 3.2 shows the proposed method can obtain close feasible solutions because the mean and maximum violation values reach a magnitude of 10^{-6} and 10^{-4}, respectively. Fig. 3.3 shows that the proposed method obtains a generator allocation strategy similar to that of the conventional optimization solver *MIPS*.

Table 3.1: The nominal values of decision variables

Test cases	Decision variables	Nominal values
IEEE-30	$\mathbf{P}_g$	0.32
	$\mathbf{Q}_g$	0.22
	$\mathbf{V}$	1.00
	$\mathbf{S}_{ij}^2$	0.03
IEEE-118	$\mathbf{P}_g$	0.80
	$\mathbf{Q}_g$	0.36
	$\mathbf{V}$	1.03
	$\mathbf{S}_{ij}^2$	0.58

Table 3.2: Feasibility evaluation of the FDPF-Dual method

Test cases	Decision variables	$\boldsymbol{\nu}$ Mean (10^{-6})	$\boldsymbol{\nu}$ Max (10^{-4})
IEEE-30	$\mathbf{P}_g$	0	0
	$\mathbf{Q}_g$	0	0
	$\mathbf{V}$	0	0
	$\mathbf{S}_{ij}^2$	0.53	0.19
IEEE-118	$\mathbf{P}_g$	0	0
	$\mathbf{Q}_g$	3.37	1.68
	$\mathbf{V}$	0	0
	$\mathbf{S}_{ij}^2$	3.10	4.50

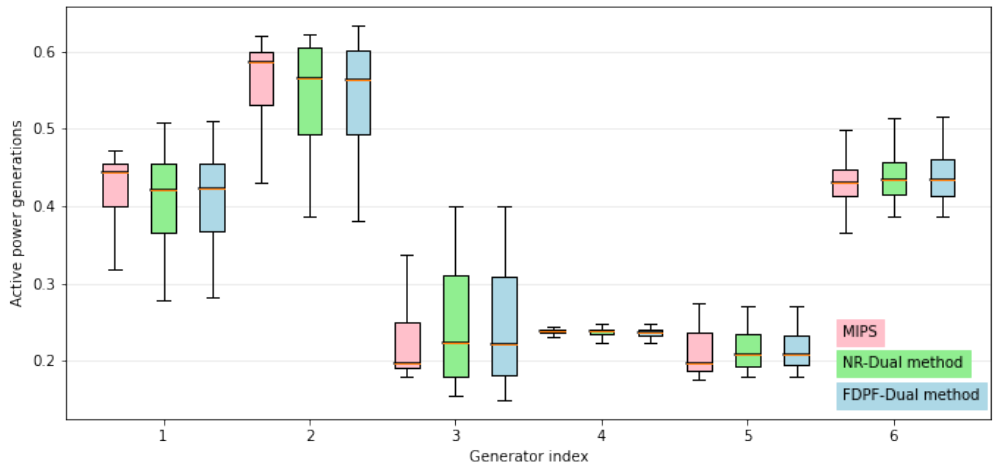


Figure 3.3: The boxplot of the active power generations on the IEEE-30 bus system.

3.2.4.6 Performance comparison with *DC3*

The training loss function of *DC3* consists of the total generation cost and inequality constraint violation penalty, as shown in Eq.(3.10). The coefficient λ_ν balances the relative importance of these two terms. For example, a smaller value of λ_ν will likely promote a solution with a lower generation cost but a larger inequality constraint violation value.

$$\mathcal{L}_{\text{DC3}} := f(\mathbf{y}, \mathbf{z}_2) + \lambda_\nu \|\boldsymbol{\nu}\|_2^2 \,. \tag{3.10}$$

Table 3.3 shows the performance of *DC3* under different values of λ_ν. For the IEEE-30 bus system, the proposed *NR-Dual* and *FDPF-Dual* methods speed up the computation by 90x and 30x than *MIPS*, respectively, with only 0.15% more expensive generation cost. With $\lambda_\nu = 1$, *DC3* achieves the cheapest generation cost, but the mean and maximum values of $\boldsymbol{\nu}$ are 15 and 11 times greater than the FDPF-Dual method.

With $\lambda_\nu = 20$, *DC3* obtains a worse solution in both feasibility and optimality than the proposed methods. In addition, the speedup factor is 68 and 220 for *NR-Dual* and *FDPF-Dual* than *MIPS* on the IEEE-118 bus system. The FDPF solver runs 3x faster than the NR solver in solving the AC-PF equations. The proposed methods achieve at least 99% feasibility rates but the generation costs are only 0.19% and 0.16% greater than *MIPS*.

Table 3.3: Performance comparison

Test cases	Methods	λ_ν	Generation cost	$\boldsymbol{\nu}$ Mean (10^{-6})	$\boldsymbol{\nu}$ Max (10^{-4})	Feasibility rate (%)	Computation time (s)
IEEE-30	*DC3*	1	0.0645	2.78	2.09	99.44	0.13
		2	0.0647	1.19	0.83	99.63	0.13
		3	0.0648	0.47	0.55	99.85	0.13
		5	0.0650	0.39	0.39	99.85	0.13
		10	0.0652	0.36	0.40	99.85	0.13
		15	0.0655	0.28	0.27	99.85	0.13
		20	0.0659	0.26	0.25	99.69	0.13
	MIPS	-	0.0646	0	0	99.99	12.05
	NR-Dual	-	0.0647	0.23	0.21	99.80	0.13
	FDPF-Dual	-	0.0647	0.18	0.19	99.78	0.39
IEEE-118	*DC3*	1	13.145	23	72	97.66	0.52
		3	13.156	8.20	31	98.48	0.52
		5	13.161	7.92	32	98.62	0.52
		10	13.174	4.27	18	98.94	0.52
		15	13.181	2.88	13	99.16	0.52
		20	13.184	2.96	13	99.11	0.52
	MIPS	-	13.137	0	0	99.95	35.33
	NR-Dual	-	13.162	1.66	9	99.21	0.52
	FDPF-Dual	-	13.158	1.45	8	99.17	0.16

3.2.4.7 Performance comparison with *NGT*

The training loss function of the *NGT* method consists of the total generation cost, inequality constraint violation penalty, and load mismatch. The NN outputs are voltage magnitudes and phase angles of all buses. Based on power balance equations, we can rebuild the active and reactive load demands of load buses, denoted by $\hat{\mathbf{x}}_d :=$

$[(\hat{\mathbf{P}}_d)_{\mathcal{N}_d}; (\hat{\mathbf{Q}}_d)_{\mathcal{N}_d}]$. The error of load mismatch can be calculated as follows:

$$\mathcal{L}_d := \|\mathbf{x}_d - \hat{\mathbf{x}}_d\|_2^2 . \tag{3.11}$$

The training loss function of the NGT method is:

$$\mathcal{L}_{\text{NGT}} := f(\mathbf{y}, \mathbf{z}_2) + \eta(1-\tau)\|\boldsymbol{\nu}\|_2^2 + \eta\tau\mathcal{L}_d , \tag{3.12}$$

where η and $\tau \in [0, 1]$ serve as the weighting parameters to balance three tasks.

In practice, a small relative error (e.g., less than 1%) of the load mismatch is acceptable [64]. However, as shown in Tables 3.4 and 3.5, the relative error of load mismatch is significantly greater than 1% on two IEEE benchmark systems. Besides, the computation times are 0.002s and 0.006s for the IEEE-30 and IEEE-118 bus systems, respectively.

Table 3.4: Performance of the NGT method on the IEEE-30 bus system

η	τ	Generation cost	$\boldsymbol{\nu}$ Mean (10^{-6})	$\boldsymbol{\nu}$ Max (10^{-4})	Feasibility rate (%)	Load mismatch (%)
5	0.2	0.0642	0.23	0.03	99.99	5.29
	0.5	0.0651	1.72	1.92	99.68	5.13
	0.8	0.0648	11.41	13.2	99.21	5.56
10	0.2	0.0646	0.20	0.25	99.95	5.55
	0.5	0.0662	1.14	1.41	99.77	5.52
	0.8	0.0664	6.33	6.78	99.38	5.22
15	0.2	0.0654	0.46	0.57	99.88	5.50
	0.5	0.0665	0.88	0.99	99.80	5.22
	0.8	0.0670	4.29	4.81	99.48	5.08

Table 3.5: Performance of the NGT method on the IEEE-118 bus system ($\tau = 0.5$)

η	Generation cost	$\boldsymbol{\nu}$ Mean (10^{-4})	$\boldsymbol{\nu}$ Max (10^{-2})	Feasibility rate (%)	Load mismatch (%)
5	8.80	2.88	2.56	79.10	140.40
10	12.71	0.43	0.25	99.20	21.93
15	12.91	0.09	0.51	98.69	18.77
20	13.07	0.00	0.20	99.50	16.94

3.3 Gradient Estimation method to Accelerate NN Training for AC-OPF

The unsupervised learning framework proposed in section 3.2 does not rely on the input-output data pairs for NN training. The optimal solution can be obtained by incorporating both the objective function and penalty term of constraint violations in the training loss function. In this context, without any guidance from the explicit targets, the NN output values can be random in the initial training iterations. Hence, in the worst-case scenario, the subsequent PF solver may fail to find any feasible solution to the AC-PF equations, which leads to training failure. In this section, we propose a semi-supervised learning framework to help the NN output find a reasonable initial solution for the subsequent PF solver.

In addition, introducing a PF solver in the learning framework brings new challenges when it comes to computing gradients of the reconstructed variables with respect to the predicted variables. Ref. [19] derives the ground truth gradient based on the implicit function theorem and shows the gradient computation involves the inverse operation of the Jacobian matrix. The computation complexity grows quadratically with the power grid size, which leads to long training time for a large-scale bus system.

A timely update of the NN by considering the new training data instances is essential to ensure its performance due to its data-driven nature. Hence, we propose a physics-informed batch-mean gradient estimation approach along with a reduced branch set to reduce the computation time.

3.3.1 Semi-supervised Learning via Data Augmentation

Fig. 3.1 depicts the proposed learning framework with the PF solver to ensure the equality constraint satisfaction. Due to its data-driven nature, the NN requires a large number of training data pairs to achieve high accuracy. Instead of relying purely on the conventional optimization solver to obtain the optimal solutions for all the demand samples, we use the pseudo-labeling technique to build a hybrid dataset, significantly reducing the data preparation time. The data generation process is given as follows. First, we use the conventional optimization solver to obtain the optimal solutions to a small portion of load demand samples. Then we employ ridge regression to learn the mapping $\mathbf{x} \mapsto \mathbf{y}$. The trained model can rapidly predict the optimal solutions for all the remaining unpaired load demand samples. These pseudo-labels may not be feasible or optimal, but they can provide essential guidance for the NN training process. Algorithm 2 presents the data preparation process.

3.3.1.1 Training Loss

The training loss function should consider the optimality and feasibility of the solution. The variable splitting scheme ensures the satisfaction of equality constraint.

Algorithm 2 AC-OPF data generation process using pseudo-labeling

Input: Load demand dataset $\mathcal{X}$.

Output: Hybrid load demand dataset $\hat{\mathcal{X}}$.

1: Use a conventional optimization solver to obtain the optimal solutions to a small portion of load demand samples in $\mathcal{X}$.
2: Utilize ridge regression to learn the mapping $\mathbf{x} \mapsto \mathbf{y}$ based on data pairs $(\mathbf{x}, \mathbf{y})$.
3: Use the trained regression model to predict pseudo labels for the remaining data samples in $\mathcal{X}$. Project the pseudo labels into their feasible range, i.e., $\mathbf{y} = \min\{\max\{\mathbf{y}, \mathbf{y}^{\min}\}, \mathbf{y}^{\max}\}$
4: Obtain $\mathbf{z}_1$ via the FDPF solver and compute $\mathbf{z}_2 = \mathbf{f}_r(\mathbf{v})$.

Thus, the training loss function consists of three parts: 1) the generation cost L_o, 2) the penalty loss of the inequality constraint violation L_c, and 2) the error between pseudo-labels and FCNN estimates L_s. The overall training loss function is designed as follows:

$$\ell(\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2) := L_c + w_o L_o + w_s L_s, \tag{3.13}$$

where w_o and w_s represent the weight parameters to balance the order of magnitude of different loss terms. The supervised learning loss L_s is given as:

$$L_s(\mathbf{P}_g, \mathbf{v}) = \|\mathbf{P}_g - \tilde{\mathbf{P}}_g\|_2 + \|\mathbf{v} - \tilde{\mathbf{v}}\|_2, \tag{3.14}$$

where $\tilde{\mathbf{P}}_g$ and $\tilde{\mathbf{v}}$ denote the pseudo-labels of the corresponding variables. Additionally, let w_v denote the weight parameter to balance the violation loss between $\mathbf{z}_2$ and $\mathbf{V}_d$.

The box constraint violation loss L_c is given as:

$$L_c(\mathbf{z}_2, \mathbf{V}_d) = l_c(\mathbf{z}_2) + w_v l_c(\mathbf{V}_d), \tag{3.15}$$

where $l_c(\mathbf{c})$ is defined as:

$$l_c(\mathbf{c}) := \|\mathrm{ReLU}(\mathbf{c} - \mathbf{c}^{\max})\|_2 + \|\mathrm{ReLU}(\mathbf{c}^{\min} - \mathbf{c})\|_2. \tag{3.16}$$

Remark (The role of weight w_v)**.** *The weight parameter value set up w_v is crucial to avoid the potential vicious cycle and guarantee feasibility. The chain rule in the backpropagation process implies:*

$$\frac{\mathrm{d}L_c(\mathbf{z}_2(\mathbf{V}_d), \mathbf{V}_d)}{\mathrm{d}\mathbf{V}_d} = \frac{\partial l_c(\mathbf{z}_2)}{\partial \mathbf{z}_2}\frac{\mathrm{d}\mathbf{z}_2}{\mathrm{d}\mathbf{V}_d} + w_v \frac{\mathrm{d}\, l_c(\mathbf{V}_d)}{\mathrm{d}\mathbf{V}_d}, \tag{3.17}$$

where $\frac{\mathrm{d}\mathbf{z}_2}{\mathrm{d}\mathbf{V}_d}$ is related to the power grid parameters, which can have significant values in the transmission power grids. Thus, the power grid parameters and w_v should have a similar order of magnitude to ensure no loss terms will be ignored in the training loss function. In the forward pass, $\mathbf{z}_2$ is dependent on $\mathbf{V}_d$ through the power flow and branch flow equations. Hence, if $\mathbf{V}_d$ is far from its upper and lower bound, the resulting $\mathbf{z}_2$ will also have a significant violation loss. In this case, more effort will be put into minimizing the first loss term while increasingly ignoring the second one, leading to a vicious cycle.

Ref. [17] shows using a pre-trained model to warm up the NN weights can help

reduce the optimality gap. Inspired by their work, we use the following supervised loss function in the first few training epochs to give the NN a warm start:

$$L_{\text{wp}}(\mathbf{P}_g, \mathbf{V}_{\bar{\mathcal{N}}_d}) = \|\mathbf{P}_g - \tilde{\mathbf{P}}_g\|_2 + w_{\text{wp}}\|\mathbf{V}_{\bar{\mathcal{N}}_d} - \tilde{\mathbf{V}}_{\bar{\mathcal{N}}_d}\|_2\,, \tag{3.18}$$

where the weight parameter w_{wp} balances the relative importance of two loss terms. The benefits are given as follows. First, in the initial stage of training, the NN output values can be pretty random. In the worst-case scenario, the PF solver may be unable to find a feasible PF solution to the NN output $\mathbf{y}$, leading to the training failure. Second, a large feasible range of the decision variables may cause heavy training time due to the extensive search space.

3.3.2 Gradient Computation

3.3.2.1 Implicit gradient computation

We need to calculate the derivative $\frac{\mathrm{d}\ell}{\mathrm{d}\mathbf{W}} = \frac{\mathrm{d}\ell}{\mathrm{d}\mathbf{y}} \times \frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\mathbf{W}}$ in the backpropagation, where $\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\mathbf{W}}$ is straightforward to obtain. Based on the chain rule, the calculation of $\frac{\mathrm{d}\ell}{\mathrm{d}\mathbf{y}}$ is given as follows:

$$\frac{\mathrm{d}\ell(\mathbf{y}, \mathbf{z}_1(\mathbf{y}), \mathbf{z}_2(\mathbf{y}, \mathbf{z}_1(\mathbf{y})))}{\mathrm{d}\mathbf{y}} = \frac{\partial \ell}{\partial \mathbf{y}} + \frac{\partial \ell}{\partial \mathbf{z}_1}\frac{\mathrm{d}\mathbf{z}_1}{\mathrm{d}\mathbf{y}} + \frac{\partial \ell}{\partial \mathbf{z}_2}\frac{\mathrm{d}\mathbf{z}_2}{\mathrm{d}\mathbf{y}}\,, \tag{3.19}$$

$$\frac{\mathrm{d}\mathbf{z}_2(\mathbf{y}, \mathbf{z}_1(\mathbf{y}))}{\mathrm{d}\mathbf{y}} = \frac{\partial \mathbf{z}_2}{\partial \mathbf{y}} + \frac{\partial \mathbf{z}_2}{\partial \mathbf{z}_1}\frac{\mathrm{d}\mathbf{z}_1}{\mathrm{d}\mathbf{y}}\,, \tag{3.20}$$

where $\frac{\partial \mathbf{z}_2}{\partial \mathbf{y}}$, $\frac{\partial \mathbf{z}_2}{\partial \mathbf{z}_1}$ and $\frac{\mathrm{d}\mathbf{z}_1}{\mathrm{d}\mathbf{y}}$ can be obtained from the nodal Jacobian matrix and branch Jacobian matrix based on the power flow and branch flow equations, respectively.

Let $\mathbf{J}^{\text{nodal}}$ denote the nodal Jacobian matrix, which collects the gradients of the active and reactive power injections with respect to the phase angles and voltage magnitudes. $\mathbf{J}^{\text{nodal}}$ be derived from Eqs.(3.1b)-(3.1c). $\mathbf{J}^{\text{nodal}}$ is composed of four blocks given as:

$$\mathbf{J}^{\text{nodal}} := \begin{bmatrix} \mathbf{J}^{P\theta} & \mathbf{J}^{PV} \\ \mathbf{J}^{Q\theta} & \mathbf{J}^{QV} \end{bmatrix} \in \mathbb{R}^{2N\times 2N}. \tag{3.21}$$

Each block is an $N \times N$ matrix. Let $\mathbf{p} = [p_1, p_2 \ldots, p_M]^\top$ and $\mathbf{q} = [q_1, q_2 \ldots, q_M]^\top$ collect the active and reactive branch flows. Let $\mathbf{v}_j$ denote the j-th element of $\mathbf{v}$ and $\odot$ represent the point-wise product. Based on the apparent branch flow equation (3.1f), we have

$$\frac{\partial\, \mathbf{s}^2}{\partial\, \mathbf{v}_j} = 2\mathbf{p} \odot \mathbf{J}^{ab}_{(:,j)} + 2\mathbf{q} \odot \mathbf{J}^{rb}_{(:,j)} \in \mathbb{R}^{M}, \tag{3.22}$$

where $\mathbf{J}^{ab} \in \mathbb{R}^{M\times 2N}$ and $\mathbf{J}^{rb} \in \mathbb{R}^{M\times 2N}$ represent the gradients of the active and reactive power flows with respect to the phase angles and voltage magnitudes, which can be derived from Eqs.(3.1d)-(3.1e). $\mathbf{J}^{ab}_{(:,j)}$ and $\mathbf{J}^{rb}_{(:,j)}$ are the j-th column of $\mathbf{J}^{ab}$ and $\mathbf{J}^{rb}$, respectively.

We can rewrite the PF equations as an implicit function $\mathbf{f}(\mathbf{y}, \mathbf{z}_1(\mathbf{y})) = \mathbf{0}$. Based

on the implicit function theorem, $\frac{\mathrm{d}\mathbf{z}_1}{\mathrm{d}\mathbf{y}}$ can be calculated by [19]:

$$\frac{\mathrm{d}\mathbf{z}_1(\mathbf{y})}{\mathrm{d}\mathbf{y}} = -(\frac{\partial \mathbf{f}}{\partial \mathbf{z}_1})^{-1}(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}) \in \mathbb{R}^{(2N_d+N_g)\times(2N_g+1)}, \tag{3.23}$$

where $\frac{\partial \mathbf{f}}{\partial \mathbf{z}_1} := \mathbf{J}_{z1}$ can be obtained as the corresponding elements of $\mathbf{J}^{\text{nodal}}$:

$$\mathbf{J}_{z1} := \begin{bmatrix} \mathbf{J}_{z1}^{P\theta} & \mathbf{J}_{z1}^{PV} \\ \mathbf{J}_{z1}^{Q\theta} & \mathbf{J}_{z1}^{QV} \end{bmatrix} \in \mathbb{R}^{(2N_d+N_g)\times(2N_d+N_g)}. \tag{3.24}$$

In addition, $\frac{\partial \mathbf{f}}{\partial \mathbf{y}} := \mathbf{J}_y$ can be calculated by:

$$\mathbf{J}_y := \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial \mathbf{P}_g} & \frac{\partial \mathbf{f}}{\partial \mathbf{V}_{\bar{\mathcal{N}}_d}} \end{bmatrix} \in \mathbb{R}^{(2N_d+N_g)\times(2N_g+1)}, \tag{3.25}$$

where $\frac{\partial \mathbf{f}}{\partial \mathbf{P}_g}$ is straightforward to compute because $\mathbf{P}_g$ only gets involved in (3.1b). Similarly, $\frac{\partial \mathbf{f}}{\partial \mathbf{V}_{\bar{\mathcal{N}}_d}}$ can be obtained as the corresponding elements of $\mathbf{J}^{\text{nodal}}$.

In addition, $\frac{\partial \mathbf{z}_2}{\partial \mathbf{P}_g} = \mathbf{0}$ because the decision variables in $\mathbf{z}_2$ depends on the voltage phasors. $\frac{\partial \mathbf{z}_2}{\partial \mathbf{y}}$ and $\frac{\partial \mathbf{z}_2}{\partial \mathbf{z}_1}$ can be obtained as the corresponding elements of $\mathbf{J}^{\text{nodal}}$ and $\mathbf{J}^{ab}/\mathbf{J}^{rb}$.

To this end, by plugging (3.23) and (3.20) into (3.19), $\frac{\mathrm{d}\ell}{\mathrm{d}\mathbf{y}}$ is obtained by

$$\frac{\mathrm{d}\ell}{\mathrm{d}\mathbf{y}} = \frac{\partial \ell}{\partial \mathbf{y}} + \frac{\partial \ell}{\partial \mathbf{z}_2}\frac{\partial \mathbf{z}_2}{\partial \mathbf{y}} + (\frac{\partial \ell}{\partial \mathbf{z}_1} + \frac{\partial \ell}{\partial \mathbf{z}_2}\frac{\partial \mathbf{z}_2}{\partial \mathbf{z}_1})(-\mathbf{J}_{z1}^{-1}\mathbf{J}_y). \tag{3.26}$$

Instead of directly calculating the matrix inverse, we can solve a system of linear equa-

tions:

$$\frac{\mathrm{d}\ell}{\mathrm{d}\mathbf{y}} = \frac{\partial \ell}{\partial \mathbf{y}} + \frac{\partial \ell}{\partial \mathbf{z}_2}\frac{\partial \mathbf{z}_2}{\partial \mathbf{y}} - \mathbf{k}^\top \mathbf{J}_y, \tag{3.27}$$

where $\mathbf{k} \in \mathbb{R}^{2N_d+N_g}$ can be calculated using:

$$\frac{\partial \ell}{\partial \mathbf{z}_1} + \frac{\partial \ell}{\partial \mathbf{z}_2}\frac{\partial \mathbf{z}_2}{\partial \mathbf{z}_1} = \mathbf{k}^\top \mathbf{J}_{z1}. \tag{3.28}$$

3.3.2.2 Proposed batch gradient estimation method

Let $\mathcal{J}_{z1} \in \mathbb{R}^{(2N_d+N_g)\times(2N_d+N_g)\times b}$ denote the tensor notation of $\mathbf{J}_{z1}$, where b refers to the batch size. In mini-batch training, we need to compute the gradient based on Eq.(3.27) at each training iteration. The computation complexity of solving Eq.(3.28) using lower–upper decomposition can be up to $\mathcal{O}(b \times (2N_d + N_g)^3)$ [84], which grows quadratically with the power grid size and gets amplified by the batch size. To relieve the computational burden, we have the following three strategies:

- Motivated by the linearized PF analysis, we propose a linearized Jacobian formulation to replace the original non-linear function computation.
- We propose a decoupled formulation of Eq.(3.28) based on the strong coupling relationships between P-θ and Q-V.
- Instead of calculating tensor $\mathcal{J}_{z1}$, we compute the mean values of a batch of samples and obtain the batch-mean Jacobian matrix.

Linearized Jacobian model: Let $\bar{v}_i$ denote the mean value of the voltage magnitude pseudo label values. In the transmission grid, the phase angle difference between two connected buses is typically small, and the voltage magnitude value is close to 1 in the per unit system [85]. Thus, we have the following assumptions:

$$\sin\theta_{ij} \approx \theta_{ij}, \tag{3.29}$$

$$\cos\theta_{ij} \approx 1, \tag{3.30}$$

$$V_i \sin\theta_{ij} \approx \bar{v}_i\theta_{ij}, \tag{3.31}$$

$$V_j \sin\theta_{ij} \approx \bar{v}_j\theta_{ij}, \tag{3.32}$$

$$V_iV_j \sin\theta_{ij} \approx \bar{v}_i\bar{v}_j\theta_{ij}, \tag{3.33}$$

$$V_iV_j \cos\theta_{ij} \approx \bar{v}_iV_j. \tag{3.34}$$

Based on Eqs.(3.29) - (3.34), the linearized Jacobian can be calculated by:

$$J_{ij}^{P\theta} = \bar{v}_i \bar{v}_j G_{ij} \theta_i - \bar{v}_i \bar{v}_j G_{ij} \theta_j - \bar{v}_i B_{ij} V_j, \tag{3.35a}$$

$$J_{ij}^{PV} = \bar{v}_i B_{ij} \theta_i - \bar{v}_i B_{ij} \theta_j + G_{ij} V_i, \tag{3.35b}$$

$$J_{ij}^{Q\theta} = -\bar{v}_i \bar{v}_j B_{ij} \theta_i + \bar{v}_i \bar{v}_j B_{ij} \theta_j - \bar{v}_i G_{ij} V_j, \tag{3.35c}$$

$$J_{ij}^{QV} = \bar{v}_i G_{ij} \theta_i - \bar{v}_i G_{ij} \theta_j - B_{ij} V_i, \tag{3.35d}$$

$$J_{ii}^{P\theta} = -\sum_{j \neq i} J_{ij}^{P\theta}, \tag{3.35e}$$

$$J_{ii}^{Q\theta} = -\sum_{j \neq i} J_{ij}^{Q\theta}, \tag{3.35f}$$

$$J_{ii}^{PV} = \sum_{j \neq i} \bar{v}_j B_{ij} \theta_i - \sum_{j \neq i} \bar{v}_j B_{ij} \theta_j + 2 G_{ii} V_i + \sum_{j \neq i} G_{ij} V_j, \tag{3.35g}$$

$$J_{ii}^{QV} = \sum_{j \neq i} \bar{v}_j G_{ij} \theta_i - \sum_{j \neq i} \bar{v}_j G_{ij} \theta_j - 2 B_{ii} V_i - \sum_{j \neq i} B_{ij} V_j. \tag{3.35h}$$

The linear equations (3.35a)-(3.35h) can be compactly written in the matrix-vector form:

$$\begin{bmatrix} J_{ij}^{P\theta} & J_{ij}^{PV} & J_{ij}^{Q\theta} & J_{ij}^{QV} \end{bmatrix}^\top = \mathbf{A}_{(ij)} \mathbf{v}_{(ij)}, \tag{3.36}$$

$$\begin{bmatrix} J_{ii}^{P\theta} & J_{ii}^{PV} & J_{ii}^{Q\theta} & J_{ii}^{QV} \end{bmatrix}^\top = \mathbf{A}_{(i)} \mathbf{v}, \tag{3.37}$$

where

$$\mathbf{v}_{(ij)} := [\theta_i \ \theta_j \ V_i \ V_j]^\top , \tag{3.38}$$

$$\mathbf{A}_{(ij)} = \begin{bmatrix} \bar{v}_i\bar{v}_j G_{ij} & -\bar{v}_i\bar{v}_j G_{ij} & 0 & -\bar{v}_i B_{ij} \\ \bar{v}_i B_{ij} & -\bar{v}_i B_{ij} & G_{ij} & 0 \\ -\bar{v}_i\bar{v}_j B_{ij} & \bar{v}_i\bar{v}_j B_{ij} & 0 & -\bar{v}_i G_{ij} \\ \bar{v}_i G_{ij} & -\bar{v}_i G_{ij} & -B_{ij} & 0 \end{bmatrix} . \tag{3.39}$$

$\mathbf{A}_{(i)} \in \mathbb{R}^{4\times 2N}$ is a sparse matrix whose i-th and $(i+N)$-th columns can be calculated by:

$$\mathbf{A}_{(i),[:,i,i+N]} = \begin{bmatrix} -\bar{v}_i \sum_{j\neq i} \bar{v}_j G_{ij} & 0 \\ \sum_{j\neq i} \bar{v}_j B_{ij} & 2G_{ii} \\ \bar{v}_i \sum_{j\neq i} \bar{v}_j B_{ij} & 0 \\ \sum_{j\neq i} \bar{v}_j G_{ij} & -2B_{ii} \end{bmatrix} , \tag{3.40}$$

and the j-th and $(j+N)$-th columns (for any bus j connected with bus i) can be

calculated by:

$$\mathbf{A}_{(i),[:,j,j+N]} = \begin{bmatrix} \bar{v}_i\bar{v}_j G_{ij} & \bar{v}_i B_{ij} \\ -\bar{v}_j B_{ij} & G_{ij} \\ -\bar{v}_i\bar{v}_j B_{ij} & \bar{v}_i G_{ij} \\ -\bar{v}_j G_{ij} & -B_{ij} \end{bmatrix}. \tag{3.41}$$

Based on Eqs. (3.1d) and (3.1e), the linearized Jacobian of the branch flow can be calculated by:

$$\begin{bmatrix} \mathbf{p}_{(ij)} \\ \mathbf{q}_{(ij)} \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{ab}_{(ij)} \\ \mathbf{A}^{rb}_{(ij)} \end{bmatrix} \mathbf{v}_{(ij)}, \tag{3.42}$$

where

$$\mathbf{p}_{(ij)} = \begin{bmatrix} \frac{\partial p_{ij}}{\partial \theta_i} & \frac{\partial p_{ij}}{\partial \theta_j} & \frac{\partial p_{ij}}{\partial V_i} & \frac{\partial p_{ij}}{\partial V_j} \end{bmatrix}^\top, \tag{3.43}$$

$$\mathbf{q}_{(ij)} = \begin{bmatrix} \frac{\partial q_{ij}}{\partial \theta_i} & \frac{\partial q_{ij}}{\partial \theta_j} & \frac{\partial q_{ij}}{\partial V_i} & \frac{\partial q_{ij}}{\partial V_j} \end{bmatrix}^\top, \tag{3.44}$$

$$\mathbf{A}_{(ij)}^{ab} = \begin{bmatrix} -\bar{v}_i\bar{v}_j G_{ij} & \bar{v}_i\bar{v}_j G_{ij} & 0 & \bar{v}_i B_{ij} \\ \bar{v}_i\bar{v}_j G_{ij} & -\bar{v}_i\bar{v}_j G_{ij} & 0 & -\bar{v}_i B_{ij} \\ \bar{v}_j B_{ij} & -\bar{v}_j B_{ij} & -2G_{ij} & G_{ij} \\ \bar{v}_i B_{ij} & -\bar{v}_i B_{ij} & G_{ij} & 0 \end{bmatrix}, \tag{3.45}$$

$$\mathbf{A}_{(ij)}^{rb} = \begin{bmatrix} \bar{v}_i\bar{v}_j B_{ij} & -\bar{v}_i\bar{v}_j B_{ij} & 0 & \bar{v}_i G_{ij} \\ -\bar{v}_i\bar{v}_j B_{ij} & \bar{v}_i\bar{v}_j B_{ij} & 0 & -\bar{v}_i G_{ij} \\ \bar{v}_j G_{ij} & -\bar{v}_j G_{ij} & 2B_{ij} & -B_{ij} \\ \bar{v}_i G_{ij} & -\bar{v}_i G_{ij} & -B_{ij} & 0 \end{bmatrix}. \tag{3.46}$$

Note that $\mathbf{A}_{(ij)}$, $\mathbf{A}_{(i)}$, $\mathbf{A}_{(ij)}^{ab}$ and $\mathbf{A}_{(ij)}^{rb}$ can be pre-computed before the NN training process, which speeds up the gradient computation time during training.

Decoupled Jacobian model

Inspired by the strong coupling relationship of P-θ and Q-V, we remove the off-diagonal blocks in $\mathbf{J}_{z1}$ and rewrite Eq. (3.28) as:

$$\frac{\partial \ell}{\partial \mathbf{z}_1} + \frac{\partial \ell}{\partial \mathbf{z}_2}\frac{\partial \mathbf{z}_2}{\partial \mathbf{z}_1} = \begin{bmatrix} \mathbf{k}_p^\top \mathbf{J}_{z1}^{P\theta} & \mathbf{k}_q^\top \mathbf{J}_{z1}^{QV} \end{bmatrix}, \tag{3.47}$$

where $\mathbf{k} := [\mathbf{k}_p; \mathbf{k}_q]$, $\mathbf{k}_p \in \mathbb{R}^{N_d+N_g}$ and $\mathbf{k}_q \in \mathbb{R}^{N_d}$. Compared to Eq. (3.28), the decoupled formulation (3.47) requires solving two smaller sets of linear systems. Thus, the computational complexity can be reduced from $\mathcal{O}((2N_d + N_g)^3)$ to $\mathcal{O}((N_d + N_g)^3)$ +

$\mathcal{O}(N_d^3)$.

Batch-mean Jacobian tensor estimate

To relieve the computational burden of the inverse operation of the Jacobian tensor, we propose a batch-mean estimation mechanism to eliminate the batch dimension. Let $\mathcal{T} \in \mathbb{R}^{2N\times b}$ and $\mathcal{T}_{ij} \in \mathbb{R}^{4\times b}$ denote the tensor expression of $\mathbf{v}$ and $\mathbf{v}_{(ij)}$, respectively. $\bar{\mathcal{T}}_{ij}$ and $\bar{\mathcal{T}}$ represent the mean values of a batch of samples. Based on Eqs.(3.36) and (3.37), we have the batch-mean Jacobian tensor estimates given as follows:

$$\mathcal{J}_{ij} \approx \mathbf{A}_{(ij)}\bar{\mathcal{T}}_{ij} \in \mathbb{R}^4,\ \mathcal{J}_{ii} \approx \mathbf{A}_{(i)}\bar{\mathcal{T}} \in \mathbb{R}^4. \tag{3.48}$$

For any data sample, the absolute errors brought about by the batch-mean replacement in $\mathcal{J}_{ij}^{P\theta}$, $\mathcal{J}_{ij}^{QV}$, $\mathcal{J}_{ii}^{P\theta}$ and $\mathcal{J}_{ii}^{QV}$ are given as:

$$e_{ij}^{P\theta} = |\bar{v}_i\bar{v}_j G_{ij}(\theta_{ij} - \bar{\Theta}_{ij}) - \bar{v}_i B_{ij}(V_j - \bar{\mathcal{V}}_j)|, \tag{3.49}$$

$$e_{ij}^{QV} = |\bar{v}_i G_{ij}(\theta_{ij} - \bar{\Theta}_{ij}) - B_{ij}(V_i - \bar{\mathcal{V}}_i)|, \tag{3.50}$$

$$e_{ii}^{P\theta} = |\sum_{j\neq i} \bar{v}_i\bar{v}_j G_{ij}(\theta_{ij} - \bar{\Theta}_{ij}) - \bar{v}_i B_{ij}(V_j - \bar{\mathcal{V}}_j)| \leq \sum_{j\neq i} e_{ij}^{P\theta}, \tag{3.51}$$

$$\begin{aligned} e_{ii}^{QV} &= |-2B_{ii}(V_i - \bar{\mathcal{V}}_i) + \sum_{j\neq i} \bar{v}_j G_{ij}(\theta_{ij} - \bar{\Theta}_{ij}) - B_{ij}(V_j - \bar{\mathcal{V}}_j)|, \\ &\leq |2B_{ii}(V_i - \bar{\mathcal{V}}_i)| + \sum_{j\neq i} |\bar{v}_j G_{ij}(\theta_{ij} - \bar{\Theta}_{ij}) - B_{ij}(V_j - \bar{\mathcal{V}}_j)|, \end{aligned} \tag{3.52}$$

where $\Theta_{ij} \in \mathbb{R}^b$ and $\mathcal{V}_i \in \mathbb{R}^b$ represent the tensor expression of θ_{ij} and V_i, respectively. Their mean values are denoted as $\bar{\Theta}_{ij}$ and $\bar{\mathcal{V}}_{\{i,j\}}$, respectively. The estimation errors

should not be significant due to the following three properties:

- The phase angle difference θ_{ij} is small, and the value of $\theta_{ij} - \bar{\Theta}_{ij}$ should be small as well. In addition, the value of conductances G_{ij} is typically smaller than that of susceptance B_{ij}. Thus, the value of $G_{ij}(\theta_{ij} - \bar{\Theta}_{ij})$ should not be significant.

- The voltage magnitude is typically operated in a small range around 1, which implies that the value of $(V_j - \bar{\mathcal{V}}_j)$ is small. Even if the value of susceptance B_{ij} is large, the multiplication value of $B_{ij}(V_j - \bar{\mathcal{V}}_j)$ can still be small.

- The diagonal entries have more significant error values than the off-diagonal entries due to the summation over $N - 1$ bus. Thanks to the sparse connection of the power grid, this summation may only contain a few non-zero values.

Reduced branch set

The gradient of the branch flows with respect to the voltage magnitude can be derived based on Eq.(3.22). As shown in Eqs.(3.1d) and (3.1e), the active and reactive branch flows of different data samples in one batch can be completely different. Thus, applying the batch-mean estimation mechanism to the branch flow Jacobian estimates may lead to significant errors. We propose a reduced branch model to relieve the computational burden of the branch Jacobian computation.

Let $\tilde{\mathcal{S}}_{ij}^2$ collect the pseudo values of the squared apparent branch flow of branch (i, j). The constraint violation will likely occur if the pseudo value is close to its upper bound. We design an indicator function $l_p(\cdot)$ to help identify those unlikely violated

constraints:

$$l_p(i,j) := \sum_{\tilde{s}_{ij} \in \tilde{\mathcal{S}}_{ij}^2} \text{ReLU}\left(\tilde{s}_{ij}^2 - \beta_b (s_{ij}^{\max})^2\right), \tag{3.53}$$

where the parameter $\beta_b \in [0,1]$ are used to quantify the likelihood of constraint violation. We can calculate $l_p(\cdot)$ for all branches in $\mathcal{M}$ and build a reduced branch set, which contains the likely violated constraints, denoted by $\mathcal{M}_r \subseteq \mathcal{M}$ with M_r elements:

$$\mathcal{M}_r = \{\, (i,j) \in \mathcal{M} \,|\, l_p(i,j) > 0 \,\}. \tag{3.54}$$

Due to the supervised training loss L_s, the NN estimates are expected to be close to the pseudo-labels. The constraints in $\mathcal{M} \setminus \mathcal{M}_r$ are unlikely to be violated during training because they are chosen based on the pseudo values. Computing the gradient for a reduced branch set can reduce the computational burden of NN training. The summary of the proposed learning semi-supervised learning framework is shown in Algorithm 3.

3.3.3 Simulation Results

The effectiveness of our proposed methods has been verified on the IEEE-118, PEGASE-1354, PEGASE-2869, and PEGASE-9241 bus systems.

3.3.3.1 Simulation setup

We generate $10,000$ data samples that are uniformly distributed over $[0.8\tilde{\mathbf{P}}_d, 1.2\tilde{\mathbf{P}}_d]$ and $[0.8\tilde{\mathbf{Q}}_d, 1.2\tilde{\mathbf{Q}}_d]$, with a training/validation/testing ratio of 7:1:2. In the data aug-

mentation process, we employ the conventional optimization solver *MIPS* [71] to generate 100 ground truth data pairs, with a training and validation ratio of 8:2. The batch size is set to 32. We implement the NN training process with the Adam optimizer based on `Pytorch 1.21.1`. `MultiStepLR` function is used for the learning rate schedule, which decays the learning rate l_r by γ when the training epochs reach the milestone. The learning rate schedule can avoid training oscillations due to large learning rates and improve training efficiency. Table 3.6 shows the hyperparameter setup for the NN training.

Table 3.6: The FCNN structure, weight parameters, training epochs, and learning rate.

Bus system	FCNN structure	w_{wp}	w_v	w_o	w_s	n_{wp}	n_{tol}	Learning rate schedule (l_r & milestone & γ)
IEEE-118	[236, 50, 236]	10	10	0.1	0.1	1	100	0.0005 & 90 & 0.2
PEGASE-1354	[2708, 50, 50, 2708]	10	100	0.01	0.01	1	100	0.0005 & 70 & 0.2
PEGASE-2869	[5738, 50, 50, 50, 5738]	10	100	0.001	0.01	1	10	0.0005 & 1 & 0.1
PEGASE-9241	[18482, 50, 50, 50, 18482]	10	100	0.0001	0.01	5	15	0.001 & 5 & 0.01

3.3.3.2 Methods for comparison

Table 3.7 shows the comparison methods. The supervised learning framework relies on the conventional optimization solver to generate the data pairs, which may lead to a long data preparation time. The data preparation time of the unsupervised learning framework is ignorable because it does not require data pairs for NN training. The data preparation time of the proposed semi-supervised learning framework is significantly less than that of the supervised learning framework. However, it can provide more guidance than the unsupervised learning framework.

The details of three decision variable splitting schemes are given as follows:

- VS_1: The NN output is $\mathbf{y}$, and the subsequent FDPF solver calculates $\mathbf{z1}$ by

Table 3.7: Methods for comparison

Framework	Variable splitting	Neural network	Method	Gradient computation method
Supervised	VS_1	FCNN	M_{FCNN} [16]	Zeroth-order estimation
			M_{STRT} [17]	Zeroth-order estimation
		Chebyshev CNN	M_{CHC} [59]	Post-processing with the PF solver
		CNN	M_{CNN} [59]	Post-processing with the PF solver
	VS_2	FCNN	M_{FCNNV} [13]	Explicit Jacobian
	VS_3	FCNN	M_{COMP} [61]	Post-processing with the PF solver
		LSTM	M_{LSTM}	Post-processing with the PF solver
Unsupervised	VS_1	FCNN	M_{DC3} [19]	Implicit Jacobian
			M_{DUAL} [86]	Implicit Jacobian
	VS_2	FCNN	M_{NGT} [64]	Explicit Jacobian
Semi-supervised	VS_1	FCNN	M_0	Batch-mean estimation with all branches
			M_1	Batch-mean estimation & reduced branch set
			M_2	Linearized Jacobian & batch-mean estimation & reduced branch set
			M_3	Decoupled Jacobian & batch-mean estimation & reduced branch set
			M_4	Linearized decoupled Jacobian & batch-mean estimation & reduced branch set

solving the inverse PF equations. The gradient of $\mathbf{z}_1$ w.r.t. $\mathbf{v}$ is complicated due to the inverse PF equations. Given voltage phasors of all buses $\mathbf{v}$, the decision variables in $\mathbf{z}_2$ can be calculated by $\mathbf{z}_2 = \mathbf{f}_r(\mathbf{v})$.

- VS_2: The NN output is $\mathbf{v}$, and the remaining decision variables in $\mathbf{P}_g$ and $\mathbf{z}_2$ are obtained using the forward power flow and branch flow equations. The gradient of $\mathbf{P}_g$ and $\mathbf{z}_2$ w.r.t. $\mathbf{v}$ is straightforward due to this explicit mapping.

- VS_3: The NN outputs are $\mathbf{v}$, $\mathbf{P}_g$ and $\mathbf{Q}_g$. Given $\mathbf{v}$, the apparent branch flows can be obtained by the branch flow equations.

The proposed work adopts VS_1 because it can ensure the power balance equations and avoid the load mismatch. The gradient computation of VS_2 is straightforward because the Jacobian can be derived explicitly from the forward power flow equations.

3.3.3.3 Evaluation criteria

We have the following criteria for a comprehensive performance evaluation:

- **Optimality:** The optimality gap is defined as $l_{\text{cost}} := \frac{C - C_o}{C_o} \times 100\%$, where C and

C_o represent the objective function obtained by the NN and `MIPS`, respectively.

- **Feasibility:** The inequality box constraint violation loss can be obtained by $l_v(\mathbf{z}_a) = \text{ReLU}\left(\mathbf{z}_a - \mathbf{z}_a^{\max}\right) + \text{ReLU}\left(\mathbf{z}_a^{\min} - \mathbf{z}_a\right)$, where $\mathbf{z}_a = [\mathbf{P}_g; P_{g,\text{ref}}; \mathbf{Q}_g; Q_{g,\text{ref}}; \mathbf{V}; \mathbf{s}^2]$. Besides, $l_v^{\max}$ and $\bar{l}_v$ denotes the maximum and mean values of $l_v(\mathbf{z}_a)$, respectively.

- **Load mismatch:** e_l denotes the ratio of the absolute error of load demand estimate to the ground truth value.

- **Computational time:** T_{train} and t_{train} denote the total training time and the average training time of each epoch, respectively. T_{opt} and T_{prop} represent the total computational time of testing samples of *MIPS* and the proposed method, respectively.

- **Storage:** The memory size of the Jacobian data.

3.3.3.4 Numerical results for the data augmentation

As shown in Table 3.8, the average std values of the decision variables are small. Ridge regression yields promising results in predicting the decision variables with small perturbations in supervised learning (cf. section 2.3). As shown in Table 3.9, the prediction errors of $\mathbf{P}_g$ and $\mathbf{V}$ are small, which indicates the pseudo values can provide practical guidance for the NN training via the supervised loss. As shown in Table 3.10, compared to supervised learning, the data preparation time of the proposed semi-supervised learning framework has been reduced significantly. Table 3.11 shows

the sizes of the reduced branch set under different values of β_b. A smaller value of β_b leads to a larger set $\mathcal{M}_r$, which implies the choice of unlikely violated constraints pretends to be more conservative and leads to a more significant computational burden. After the heuristic tuning process, we set $\beta_b = 0.7$ in the simulations.

Table 3.8: The average std values and the ridge regression parameters of the decision variables.

Bus system	std $(\mathbf{P}_g)$	std $(\mathbf{V}_{\overline{\mathcal{N}}_d})$	α_p	α_v
IEEE-118	0.032	0.002	0.01	0.1
PEGASE-1354	0.067	0.001	1	1
PEGASE-2869	0.082	0.001	10	10
PEGASE-9241	0.062	0.003	100	500

Table 3.9: The ℓ_2 norm of prediction errors of the ridge regression method.

Bus system	$\ell_2(\mathbf{P}_g)$	$\ell_2(\mathbf{Q}_g)$	$\ell_2(\mathbf{V})$	$\ell_2(\boldsymbol{\theta})$
IEEE-118	0.165	0.278	0.018	0.219
PEGASE-1354	1.828	1.235	0.042	0.691
PEGASE-2869	4.279	2.061	0.083	2.993
PEGASE-9241	5.599	4.615	0.406	8.825

Table 3.10: Data preparation time.

Bus system	Learning framework	Time
IEEE-118	Supervised	0.2h
	Unsupervised	0.02s
	Semi-supervised	0.2min
PEGASE-1354	Supervised	2.2h
	Unsupervised	0.3s
	Semi-supervised	3min
PEGASE-2869	Supervised	6.4h
	Unsupervised	0.6s
	Semi-supervised	9min
PEGASE-9241	Supervised	12.2h
	Unsupervised	0.9s
	Semi-supervised	20.5min

Table 3.11: The size of reduced branch set under different β_b. The third column shows the ratio of M_r to M.

Bus system	β_b	M_r	M_r/M
IEEE-118	0.9	29	0.15
	0.7	33	0.17
	0.5	55	0.29
	0.3	94	0.50
PEGASE-1354	0.9	18	0.009
	0.7	42	0.02
	0.5	113	0.05
	0.3	245	0.12
PEGASE-2869	0.9	30	0.006
	0.7	51	0.01
	0.5	101	0.02
	0.3	243	0.05
PEGASE-9241	0.9	37	0.002
	0.7	73	0.004
	0.5	134	0.008
	0.3	356	0.022

3.3.3.5 Performance comparison for NN

Table 3.12 shows the performance comparison results. On the PEGASE-1354/-2869/-9241 bus systems, the proposed methods achieve the best performance in feasibility. The proposed methods achieve lower generation cost than M_{FCNN} and M_{DC3}. On the PEGASE-9241 bus system, without data pair guidance, the training of M_{DC3} and M_{DUAL} based on the unsupervised learning framework fail because the follow-up FDPF solver cannot find any PF feasible solution. The constraint violation loss of the proposed methods is much smaller than M_{STRT} without sacrificing much in optimality. M_{FCNNV} and M_{NGT} adopt VS_2, which results in significant load mismatches. M_{COMP} and M_{LSTM} avoid load mismatches by using the FDPF solver in post-processing but leads to significant inequality constraint violations. In addition, the performance of

M_0 achieves comparable results as M_1, which implies the unlikely violated constraints identified by the indicator function do not affect the optimal solution. M_4 achieves comparable results as M_1, which indicates the proposed gradient estimation method based on the linearized and decoupled Jacobian matrix does not significantly impact the training accuracy.

As shown in Table 3.13, M_4 achieves comparable training time of each epoch as the zeroth-order method. Compared to M_{DC3} using the ground truth Jacobian tensor, the proposed gradient estimation methods require much less storage and computational time. On the IEEE-118, PEGASE-1354 and PEGASE-2869 bus systems, the speedup ratios of M_4 to M_{DC3} of t_{train} are 7x, 12x, and 18x, respectively. The proposed methods on the PEGASE-9241 bus system can complete data preparation and training in 9 hours and achieve daily NN updates. As shown in Table 3.14, the total computational time of testing samples of the proposed method is much less than that of the conventional optimization solver. Thus, the proposed methods have shown significant advantages in the real-time energy market for faster market clearing time.

Table 3.12: Comparison Results: Feasibility and Optimality. Training failure (✗) is indicated when the FDPF solver cannot find feasible PF solutions. The winners of methods without load mismatch are highlighted in bold.

Cases	Evaluation metrics	M_{FCNN}	M_{STRT}	M_{FCNNV}	M_{CHC}	M_{CNN}	M_{COMP}	M_{LSTM}	M_{DC3}	M_{DUAL}	M_{NGT}	M_0	M_1	M_2	M_3	M_4
118	$l_v^{\max}(10^{-1})$	0.33	**0.01**	0.89	0.46	3.05	2.25	0.53	**0.01**	0.05	0.00	0.04	0.02	0.03	0.10	0.06
	$\bar{l}_v(10^{-4})$	0.68	**0.02**	5.45	1.60	7.84	11.2	2.73	**0.02**	0.13	0.00	0.07	0.04	0.05	0.18	0.12
	l_{cost}	0.88	0.30	0.00	0.00	0.00	0.04	0.02	0.27	0.04	-1.22	0.32	0.25	0.26	0.49	0.54
	e_l (%)	0	0	10	0	0	0	0	0	0	10	0	0	0	0	0
1354	$l_v^{\max}(10^{-1})$	5.8	3.8	26.4	10.1	31.1	42.4	16.3	4.8	3.7	4.0	1.6	**1.3**	**1.3**	**1.3**	1.4
	$\bar{l}_v(10^{-4})$	8.6	2.8	26.2	5.1	14.7	38.0	18.2	2.3	3.0	3.4	0.63	0.44	**0.41**	0.46	0.44
	l_{cost}	0.98	0.03	0.08	0.00	0.00	0.00	0.00	0.76	0.80	-2.99	0.05	0.05	0.05	0.06	0.06
	e_l (%)	0	0	12	0	0	0	0	0	0	22	0	0	0	0	0
2869	$l_v^{\max}(10^{-1})$	13.5	6.6	72.3	32.6	66.4	72.5	134.6	6.1	7.0	3.6	4.1	**3.6**	**3.6**	4.1	3.9
	$\bar{l}_v(10^{-4})$	43.4	6.7	22.0	9.9	13.2	19.8	78.2	1.8	5.0	1.5	1.3	**0.9**	1.1	1.1	1.2
	l_{cost}	0.91	0.08	0.12	0.00	0.00	0.00	0.01	0.80	1.49	-2.24	0.04	0.04	0.04	0.09	0.09
	e_l (%)	0	0	544	0	0	0	0	0	0	284	0	0	0	0	0
9241	$l_v^{\max}(10^{-1})$	✗	165.2	52.6	18095	-	129.2	567.7	✗	✗	2.2	33.6	**23.6**	27.7	29.2	28.4
	$\bar{l}_v(10^{-4})$		70.0	10.2	1455		41.8	74.9			4.6	5.4	4.0	**3.9**	4.8	4.0
	l_{cost}		0.01	0.05	0.03		0.12	0.00			-3.13	0.03	0.03	0.03	0.03	0.03
	e_l (%)		0	484	0		0	0			128	0	0	0	0	0

Table 3.13: The training time and the memory size of the gradient data.

System	Gradient calculation	Method	Storage	t_{train}	T_{train}
IEEE-118	zeroth-order	M_{FCNN}	0.02MB	2s	6min
		M_{STRT}		2s	6min
	Jacobian	M_{DC3}	14.4MB	29s	48min
	batch mean	M_0	0.58MB	17s	28min
		M_1	0.47MB	11s	18min
		M_2		4s	7min
		M_3		11s	18min
		M_4		4s	7min
PEGASE-1354	zeroth-order	M_{FCNN}	0.13MB	2.1min	17.5h
		M_{STRT}		1.3min	4.3h
	Jacobian	M_{DC3}	1.87GB	16.0min	6.6h
	batch mean	M_0	0.06GB	3.5min	5.8h
		M_1	0.05GB	2.4min	4.0h
		M_2		1.5min	2.5h
		M_3		2.2min	3.6h
		M_4		1.3min	2.1h
2869	zeroth-order	M_{FCNN}	0.260MB	7.9min	26.3h
		M_{STRT}		6.1min	10.1h
	Jacobian	M_{DC3}	8.432GB	89.0min	14.8h
	batch mean	M_0	0.266GB	10.9min	101min
		M_1	0.263GB	8.1min	75min
		M_2		6.0min	56min
		M_3		7.0min	62min
		M_4		4.9min	47min
9241	zeroth-order	M_{FCNN}	0.73MB	✗	✗
		M_{STRT}		39.3 min	10.1h
	Jacobian	M_{DC3}	87.457GB	✗	✗
	batch mean	M_0	2.745GB	49.5min	8.2h
		M_1	2.737GB	44.5min	7.4h
		M_2		32.7min	5.4h
		M_3		32.4min	5.4h
		M_4		31.1min	5.1h

3.3.3.6 Convergence analysis

Fig. 3.4 shows the training loss evolution process of different gradient estimation methods. The proposed method achieves a comparable convergence rate as

Table 3.14: The testing time comparison results.

System	T_{prop}	T_{opt}
118	0.3s	3.4min
PEGASE-1354	10.4s	37.7min
PEGASE-2869	50.2s	109.7min
PEGASE-9241	361.7s	209.1min

the ground truth method. The convergence performance of the zeroth-order gradient method is the worst because it cannot provide accurate gradient descent directions for the NN weight updates. We implement 5 simulation runs independently, and Fig. 3.5 and Fig. 3.6 show the training loss evolution process of M_4 on the PEGASE-1354 and IEEE-118 bus systems, respectively. The oscillations are due to a large learning rate instead of adopting the proposed estimated gradients in training. The training loss stabilizes after reducing the learning rate and finally converges to a better solution. As shown in Fig. 3.7, a large learning rate leads to oscillations, while a small learning rate leads to a slow convergence rate. The adoption of the learning rate schedule strategy can help avoid overshooting the local minimum and improve training efficiency.

3.4 Summary

This chapter proposes a novel end-to-end unsupervised learning-based framework for the AC-OPF analysis. Given load demands, the framework can rapidly yield a high-quality optimal solution. Power balance and branch flow equations are guaranteed to be satisfied using the decision variable splitting. The augmented Lagrangian function is adopted as the training loss function. Numerical results show that our approach

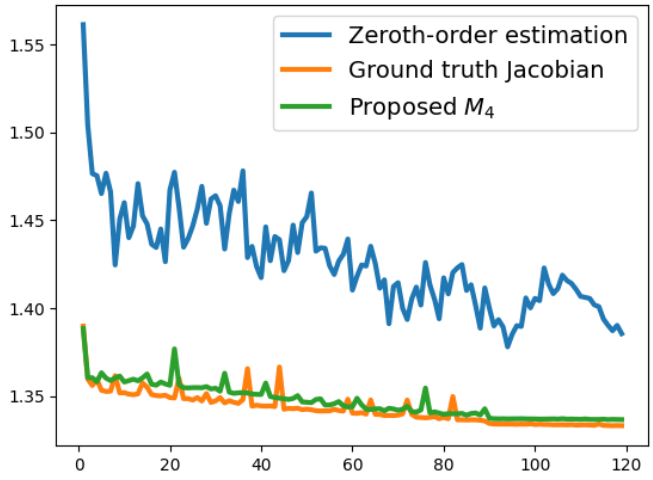


Figure 3.4: The training loss trajectory of different gradient estimation methods on the IEEE-118 bus system. The learning rate is set to 0.0005 for the initial 90 epochs, then reduced to 0.0001 for the last 30 epochs.

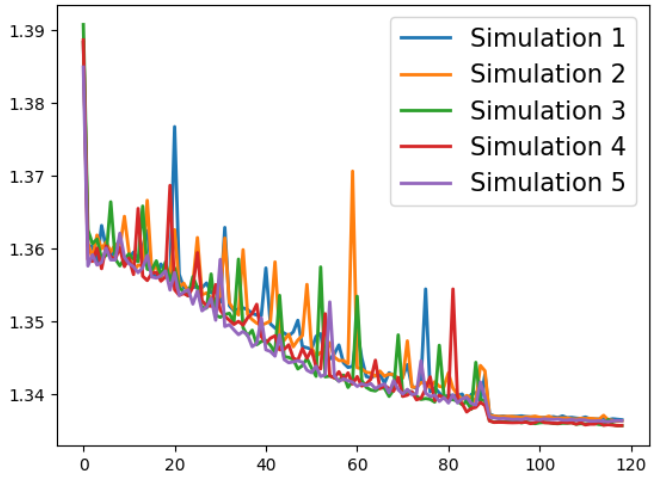


Figure 3.5: The total training loss trajectory of the proposed method M_4 over epochs on the IEEE-118 bus system. The learning rate is 0.0005 in the first 90 epochs and 0.0001 in the last 30 epochs.

outperforms two existing unsupervised learning-based works.

Next, we propose physics-informed gradient estimation methods based on the semi-supervised learning framework to speed up NN training for large-scale power grids.

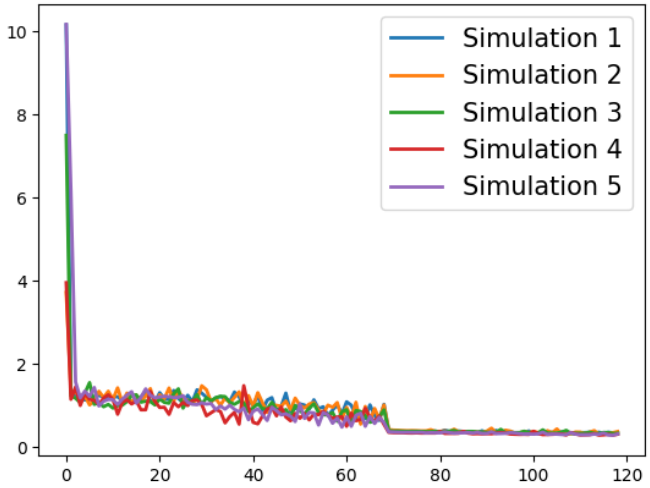


Figure 3.6: The total training loss evolution process of the proposed method M_4 over epochs on the PEGASE-1354 bus system. The learning rate is 0.0005 in the first 70 epochs and 0.0001 in the last 50 epochs.

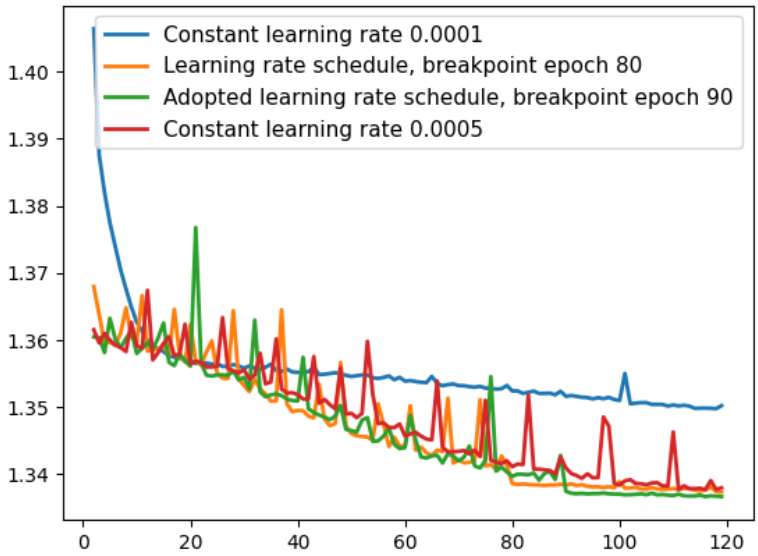


Figure 3.7: The total training loss trajectory of the proposed method M_4 over epochs using different learning rate schedules on the IEEE-118 bus system. The plot starts from the second epoch for better demonstration.

By utilizing the data augmentation process aided by ridge regression, the data pair preparation time has been significantly reduced compared to the supervised learning

framework. The proposed batch-mean linearized decoupled gradient estimation methods, along with the reduced branch set, relieve the computational burden of computing the ground truth Jacobian tensor in NN training. The NN training can be completed within one day, which enables timely and frequent NN updates to accommodate new instances. The trained NN will serve as a rapid online AC-OPF solver to obtain the optimal system operating states in the real-time energy market.

Algorithm 3 Semi-supervised OPF learning framework utilizing batch-mean gradient estimation

Input: Load demand dataset $\mathcal{X}$.
Output: The trained FCNN.

Construct the hybrid load demand dataset $\hat{\mathcal{X}}$ using the Algorithm 1. ▷ `Pseudo-labeling`
Calculate the indicator function Eq.(3.53) to identify the reduced branch set $\mathcal{M}_r$.
for episode $e = 1, 2, \ldots, n_{\text{tol}}$ **do**
▷ `The variable splitting scheme`
    Let the FCNN predict the decision variables $\mathbf{y}$, $\mathbf{z}_1$, and $\mathbf{z}_2$; cf. Fig. 3.1.
▷ `The branch set`
    if the method M_0 is adopted **then**
        Calculate the Jacobian for all branches in $\mathcal{M}$ using Eq. (3.22).
    else if for the method M_1-M_4 **then**
        Calculate the Jacobian for branches in the reduced branch set $\mathcal{M}_r$ using Eq. (3.22).
    end if
▷ `The Jacobian computation scheme`
    if for the method M_0, M_1, or M_3 **then**
        Compute the nodal Jacobian matrix (3.21) based on the PF equations (3.1b)-(3.1c).
    else if for the method M_2 or M_4 **then**
        Calculate the linearized Jacobian matrix using Eqs. (3.36) and (3.37).
    end if
    Calculate batch-mean gradient using the Eq. (3.48)
▷ `The training loss`
    if $e \leq n_{\text{wp}}$ **then**:
        Calculate the training loss function Eq. (3.18) for the warm-up epochs.
    else
        Calculate the training loss function Eq. (3.13).
    end if
▷ `The decoupled formulation`
    if for the method M_0, M_1, or M_2 **then**
        Calculate the derivative $\frac{\mathrm{d}\ell}{\mathrm{d}\mathbf{y}}$ using Eq. (3.27).
    else if for the method M_3 or M_4 **then**
        Compute the derivative $\frac{\mathrm{d}\ell}{\mathrm{d}\mathbf{y}}$ using the decoupled formulation Eq. (3.47).
    end if
    Update the weights of the FCNN through backpropagation until the training loss converges.
end for

Chapter 4

Conclusion and Future work

4.1 Conclusion

In this thesis, we propose two deep learning-based approaches in place of conventional PF solvers to solve the AC-PF equations rapidly. With the aid of NN, the cumulative computational time of solving lots of power injection samples can be significantly reduced in PPF analysis. In addition, we propose two deep learning-based approaches to rapidly solve the AC-OPF problem, which is appealing in real-time energy markets.

In the first work, we propose a novel residual learning-based framework with three designed initialization schemes. A shortcut connection linear layer between the input and the output is introduced to the vanilla MLP structure. Hence, the MLP aims to approximate the non-linear residuals to the linearized AC-PF equations instead of learning the original non-linear functions. We propose three initialization schemes to

utilize the linear property of the AC-PF equations. Two model-based methods require knowledge of power grid topology and line parameters. The data-driven method can address the missing power grid parameter issue because it achieves promising performance by capturing the linearity based on the historical dataset. Simulation results show that the proposed NN framework with the initialization schemes can improve the estimation accuracy of voltage phasors. In addition, the residual learning-based framework achieves a better convergence property than the vanilla MLP, which has significant advantages when the training time is limited.

However, the numerical results show the advantages of the proposed methods are more evident in estimating voltage phasors than branch flows. Hence, in the second work, we propose a hybrid learning strategy to improve the estimation accuracy of voltage magnitudes. Specifically, we split the load buses into two subsets according to the fluctuations of the historical voltage magnitudes. We employ ordinary least-square linear regression for the load buses with smaller std values, while we use the MLP for the load buses with larger std values. In addition, we incorporate the phase angle difference estimates error into the training loss, which improves the estimation accuracy for branch flows. Numerical results demonstrate the effectiveness of the proposed methods in improving the voltage phasor and branch flow estimates.

In the third work, we propose an end-to-end unsupervised learning framework to solve the AC-OPF problem rapidly. The power balance and branch flow equations are strictly satisfied by incorporating the PF solver into the learning framework. The augmented Lagrangian function is used as the training loss to ensure feasibility and

optimality, which consists of the generation cost and the inequality constraint violation. The Lagrange multiplier is updated according to the inequality constraint violation value during training. Simulation results show that the proposed work can quickly yield highly feasible solutions with negligible optimality losses.

Integrating the PF solver into the learning framework leads to a heavy computation burden when it comes to gradient computation. In the fourth work, we propose a semi-supervised learning framework with novel batch-mean gradient estimation methods to improve training efficiency. The pseudo-labeling technique based on ridge regression is utilized to build a hybrid dataset, which can significantly reduce data pair preparation time. In addition, motivated by the linear properties of the AC-PF equations, we propose novel physics-informed batch-mean gradient estimation approaches to accelerate the backpropagation process. We utilize a reduced branch set to alleviate the computation complexity arising from the branch flow gradient calculations. The simulation results show that the proposed learning framework can rapidly yield feasible and near-optimal solutions to AC-OPF problems. Moreover, the proposed gradient estimation method achieves a comparable convergence performance as the ground truth Jacobian gradient but requires much less training time and storage. The proposed method enables timely and frequent updates of the NNs, which is essential to ensure the NN performance due to its data-driven nature.

4.2 Future Work

Renewable energy sources (RESs) are more environmentally friendly and cheaper than conventional energy sources such as coal and oil. However, RESs such as wind and solar are sensitive to ambient conditions, which brings significant fluctuations to the power grid operating states. The high penetration of RESs in the power grid has presented more challenges to the independent system operators. A battery energy storage system (BESS) is an advanced technique that allows renewable energy to be stored and released flexibly. BESS is essential in integrating RESs into the power grid because it can effectively relieve renewable energy uncertainty. The penetration of RES and battery leads to a multi-period OPF problem, which considers the dynamics of the battery and the real-time prediction of renewable energy generation. Hence, we will extend the single-period AC-OPF problem 3.1 to a multi-period AC-OPF problem in future work.

4.2.1 Literature Review

The multi-period AC-OPF problem can be formulated as a mixed integer non-convex programming problem and solved by the conventional optimization solver. An SOCP relaxation is proposed to find the globally optimal solution, which aims to solve the multi-period OPF problems for large-scale power grids in a reasonable time [87]. Ref. [88] proposes a spatially distributed algorithm in parallel to reduce the computation time of the multi-period OPF problem. Ref. [89] shows the high-memory GPU can solve a multi-period OPF instance with more than 10 million variables can be

solved in less than 10 minutes. Ref. [90] develops a sequential convex programming procedure to tighten the SOCP relaxation gap. To address the renewable energy uncertainty, a scenario reduction method is employed to reduce the computation complexity brought about by the enormous scenarios of RESs [91]. However, these optimization-based methods cannot rapidly solve the multi-period AC-OPF problem under renewable energy uncertainty, which prevents usage in the real-time energy market.

There is an increasing interest in utilizing deep learning to solve the multi-period AC-OPF in real-time applications. Ref. [92] formulates the multi-period AC-OPF problem as a sequence-to-sequence learning problem and employs a long short-term memory recurrent neural network to obtain the optimal power generation schedule. To cope with the computationally demanding scenario, a generative adversarial network is adopted to select representative periods for renewable energy generations [93]. A sampling technique based on the determinantal point process is used to find a small set of significant samples to enhance the computational efficiency [94]. However, the major drawback of these deep learning-based approaches is that they do not obtain the optimal solution that is responsive to renewable energy forecasting errors. In other words, they rely on the forecast values to optimize the multi-period OPF problem in a static way.

Reinforcement learning has shown promising capability in solving stochastic optimization problems and has been widely used in the power grid control domain. The multi-period AC-OPF problem can be formulated as a Markov decision process, and the RL agent can be used to optimize the decision variables under the system uncertainty. Ref. [95] utilizes the RL agent to determine the energy storage (ES) installations under

the stochastic RESs, load demands, and storage prices. Ref. [96] employs the RL agent to guide the expert to solve the real-time scheduling problem in the high-penetrated renewable power grid. Inspired by the primal-dual method, Ref. [97] utilizes the RL agent to determine the power generation output and the battery's operating status in the stochastic dynamic OPF. Their proposed method archives a higher feasibility rate than the unsupervised learning method M_{DC3} and the supervised learning method M_{FCNN}. However, the training reward curve fluctuates up and down in a wide range, which indicates an unstable training process and results in many local minima. Ref. [98] shows RL outperforms the model predictive controller in the critical load restoration process under imperfect renewable forecast errors. Therefore, in future work, we plan to utilize the RL agent to optimize the multi-period AC-OPF to investigate the effect of renewable energy forecast errors.

4.2.2 Problem Formulation

This section details the objective function and the operational constraints of the multi-period AC-OPF problem formulation.

4.2.2.1 Objective function

The objective function Eq.(4.1) consists of the total generation cost of the conventional generators and the line loss from battery charging and discharging [97]. Let $c_i(\cdot)$ denote the cost function of the generator at bus i. $P^{\text{g}}_{i,t}$ and $Q^{\text{g}}_{i,t}$ are the active and reactive power generation outputs of generator bus i. $P^{\text{c}}_{i,t}$ and $P^{\text{dis}}_{i,t}$ denote the

charging and discharging power values of the energy storage (ES) unit at bus i and time t. η_i^{c} and η_i^{dis} refer to the charging and discharging efficiency.

$$\min \sum_t \sum_i \left(c_i P_{i,t}^{\mathrm{g}} + (1-\eta_i^{\mathrm{c}}) P_{i,t}^{\mathrm{c}} + (\frac{1}{\eta_i^{\mathrm{d}}} - 1) P_{i,t}^{\mathrm{dis}} \right) . \tag{4.1}$$

4.2.2.2 Conventional generators

Let $\bar{P}_i^{\mathrm{g}}$ and $\bar{S}_i^{\mathrm{g}}$ denote the maximum active and apparent power capacities of the generator at bus i, respectively. Eq.(4.2a) shows the active power constraint, and Eq.(4.2b) depicts the apparent power capacity limit.

$$0 \leq P_{i,t}^{\mathrm{g}} \leq \bar{P}_i^{\mathrm{g}} , \tag{4.2a}$$

$$(P_{i,t}^{\mathrm{g}})^2 + (Q_{i,t}^{\mathrm{g}})^2 \leq (\bar{S}_i^{\mathrm{g}})^2 . \tag{4.2b}$$

4.2.2.3 Renewable energy sources

Let $P_{i,t}^{\mathrm{res}}$ and $Q_{i,t}^{\mathrm{res}}$ denote the active and reactive power generation outputs of the RES installed at bus i at time t. $\hat{P}_{i,t}^{\mathrm{res}}$ denotes the forecast value of the active power output. Eq.(4.3a) and Eq.(4.3b) show the operational limits of the active and reactive power output from the RES at the i-th bus, respectively.

$$0 \leq P_{i,t}^{\mathrm{res}} \leq \hat{P}_{i,t}^{\mathrm{res}} , \tag{4.3a}$$

$$(P_{i,t}^{\mathrm{res}})^2 + (Q_{i,t}^{\mathrm{res}})^2 \leq (\bar{S}_i^{\mathrm{res}})^2 \tag{4.3b}$$

4.2.2.4 Energy storage

$P^{\text{es}}_{i,t}$ and $\bar{P}^{\text{es}}_{i}$ represent the active power exchange value and its maximum capacity, respectively. $\text{SoC}_{i,t}$ denotes the value of the state of charge (SoC). C_i represents the maximum power capacity of the ES unit. Eq.(4.4a) limits the active power exchange value of the ES unit at bus i. Eq.(4.4b) indicates the battery cannot be charged and discharged simultaneously. The apparent power capacity limit is given in Eq.(4.4c). The dynamic constraint of SoC and its operational limit are shown in Eq.(4.4d).

$$0 \le P^{\text{c}}_{i,t} \le \bar{P}^{\text{es}}_{i}\,, \quad 0 \le P^{\text{dis}}_{i,t} \le \bar{P}^{\text{es}}_{i}\,, \tag{4.4a}$$

$$P^{\text{c}}_{i,t} P^{\text{dis}}_{i,t} = 0\,, \quad P^{\text{es}}_{i,t} = P^{\text{dis}}_{i,t} - P^{\text{c}}_{i,t}\,, \tag{4.4b}$$

$$(P^{\text{es}}_{i,t})^2 + (Q^{\text{es}}_{i,t})^2 \le (\bar{S}^{\text{es}}_{i})^2\,, \tag{4.4c}$$

$$\underline{\text{SoC}}_i \le \text{SoC}_{i,t} \le \overline{\text{SoC}}_i \quad \text{SoC}_{i,t} = \text{SoC}_{i,t-1} + \left(\eta^{\text{c}}_i P^{\text{c}}_{i,t} - \frac{P^{\text{dis}}_{i,t}}{\eta^{\text{d}}_i}\right) \times \frac{\Delta t}{\text{C}_i}. \tag{4.4d}$$

4.2.2.5 Power flow constraints

Let $P^{\text{d}}_{i,t}$ and $Q^{\text{d}}_{i,t}$ represent the active and reactive load demands of the i-th bus at time t. $V_{i,t}$ is the voltage magnitude and $\theta_{ij,t}$ is the phase angle difference between bus i and bus j. $P_{ij,t}$ and $Q_{ij,t}$ are the active and reactive branch flows. Eqs.(4.5a) and (4.5b) show the active and reactive power balance equations. Eqs.(4.5c), (4.5d) and (4.5e) are branch flow equations. Eq.(4.5f) refers to the apparent branch flow limit.

Eq.(4.5g) implies the phase angle of the slack bus is set to 0.

$$P^{\mathrm{g}}_{i,t} - P^{\mathrm{d}}_{i,t} = V_{i,t} \sum_{j=1}^{N} V_{j,t}(G_{ij}\cos\theta_{ij,t} + B_{ij}\sin\theta_{ij,t}) \tag{4.5a}$$

$$Q^{\mathrm{g}}_{i,t} - Q^{\mathrm{d}}_{i,t} = V_{i,t} \sum_{j=1}^{N} V_{j,t}(G_{ij}\sin\theta_{ij,t} - B_{ij}\cos\theta_{ij,t}) \tag{4.5b}$$

$$P_{ij,t} = -G_{ij}V^2_{i,t} + V_{i,t}V_{j,t}(G_{ij}\cos\theta_{ij,t} + B_{ij}\sin\theta_{ij,t}) \tag{4.5c}$$

$$Q_{ij,t} = B_{ij}V^2_{i,t} + V_{i,t}V_{j,t}(G_{ij}\sin\theta_{ij,t} - B_{ij}\cos\theta_{ij,t}) \tag{4.5d}$$

$$P^2_{ij,t} + Q^2_{ij,t} = |S_{ij,t}|^2\,, \tag{4.5e}$$

$$|S_{ij,t}|^2 \le (S^{\max}_{ij,t})^2\,, \tag{4.5f}$$

$$\theta_{\mathrm{ref},t} = 0 \tag{4.5g}$$

4.2.2.6 Voltage limit

The voltage magnitude should be operated within a desired range:

$$V_i^{\min} \le V_i \le V_i^{\max} \tag{4.6a}$$

4.2.2.7 Multi-period AC-OPF probelm

Let $X_{i,t} := \{P^{\mathrm{g}}_{i,t}, Q^{\mathrm{g}}_{i,t}, P^{\mathrm{res}}_{i,t}, Q^{\mathrm{res}}_{i,t}, P^{\mathrm{c}}_{i,t}, P^{\mathrm{dis}}_{i,t}, Q^{\mathrm{es}}_{i,t}, P_{ij,t}, Q_{ij,t}, V_{i,t}\theta_{i,t}\}$ collect the decision variables. The multi-period AC-OPF problem can be expressed as:

$$\boxed{\max_{\{X_{i,t}\}_{i\in\mathcal{N},\,t\in\mathcal{T}}} \ (4.1)\,, \quad \text{s.t.} \quad (4.2a) - (4.6a).} \tag{4.7}$$

References

[1] P. Chiradeja and R. Ramakumar, "An approach to quantify the technical benefits of distributed generation," *IEEE Transactions on Energy Conversion*, vol. 19, no. 4, pp. 764–773, 2004.

[2] G. E. Constante-Flores and M. Illindala, "Data-driven probabilistic power flow analysis for a distribution system with renewable energy sources using monte carlo simulation," in *2017 IEEE/IAS 53rd Industrial and Commercial Power Systems Technical Conference (ICPS)*, pp. 1–8, 2017.

[3] A. M. Leite da Silva and A. M. de Castro, "Risk assessment in probabilistic load flow via monte carlo simulation and cross-entropy method," *IEEE Transactions on Power Systems*, vol. 34, no. 2, pp. 1193–1202, 2019.

[4] Y. Liu, N. Zhang, Y. Wang, J. Yang, and C. Kang, "Data-driven power flow linearization: A regression approach," *IEEE Transactions on Smart Grid*, vol. 10, no. 3, pp. 2569–2580, 2019.

[5] Y. Liu, Y. Wang, N. Zhang, D. Lu, and C. Kang, "A data-driven approach to linearize power flow equations considering measurement noise," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2576–2587, 2020.

[6] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[7] K. Chen and Y. Zhang, "Physics-guided residual learning for probabilistic power flow analysis," *IEEE Access*, vol. 11, pp. 90309–90321, 2023.

[8] K. Chen and Y. Zhang, "Variation-cognizant probabilistic power flow analysis via multi-task learning," in *2022 IEEE Power and Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pp. 1–5, 2022.

[9] S. H. Low, "Convex relaxation of optimal power flow—part ii: Exactness," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 2, pp. 177–189, 2014.

[10] Z. Yang, H. Zhong, Q. Xia, and C. Kang, "Solving opf using linear approximations: fundamental analysis and numerical demonstration," *IET Generation, Transmission and Distribution*, vol. 11, no. 17, pp. 4115–4125, 2017.

[11] M. Li, S. Kolouri, and J. Mohammadi, "Learning to optimize distributed optimization: Admm-based dc-opf case study," in *2023 IEEE Power and Energy Society General Meeting (PESGM)*, pp. 1–5, 2023.

[12] K. Baker, "Solutions of dc opf are never ac feasible," 2019.

[13] W. Huang, X. Pan, M. Chen, and S. H. Low, "Deepopf-v: Solving AC-OPF problems efficiently," *IEEE Transactions on Power Systems*, vol. 37, pp. 800–803, 2021.

[14] 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 Transactions on Power Systems*, vol. 36, no. 1, pp. 346–354, 2021.

[15] J. Rahman, C. Feng, and J. Zhang, "A learning-augmented approach for ac optimal power flow," *International Journal of Electrical Power and Energy Systems*, vol. 130, 9 2021.

[16] X. Pan, M. Chen, T. Zhao, and S. H. Low, "Deepopf: A feasibility-optimized deep neural network approach for ac optimal power flow problems," *IEEE Systems Journal*, vol. 17, no. 1, pp. 673–683, 2023.

[17] Z. Wang, J.-H. Menke, F. Schäfer, M. Braun, and A. Scheidler, "Approximating multi-purpose ac optimal power flow with reinforcement trained artificial neural network," *Energy and AI*, vol. 7, p. 100133, 2022.

[18] K. Chen, S. Bose, and Y. Zhang, "Unsupervised deep learning for ac optimal power flow via lagrangian duality," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 5305–5310, 2022.

[19] P. Donti, D. Rolnick, and J. Z. Kolter, "Dc3: A learning method for optimization with hard constraints," in *International Conference on Machine Learning (ICML)*, 2021.

[20] B. Stott and O. Alsac, "Fast decoupled load flow," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-93, no. 3, pp. 859–869, 1974.

[21] H. Yu and A. Sano, "Semi-supervised learning and data augmentation for wearable-based health monitoring system in the wild," in *NeurIPS 2022 Workshop on Learning from Time Series for Health*, 2022.

[22] K. Chen, S. Bose, and Y. Zhang, "Physics-informed gradient estimation for accelerating deep learning based ac-opf," 2025.

[23] D. Deka and S. Misra, "Learning for dc-opf: Classifying active sets using neural nets," in *2019 IEEE Milan PowerTech*, pp. 1–6, 2019.

[24] A. Robson, M. Jamei, C. Ududec, and L. Mones, "Learning an optimally reduced formulation of opf through meta-optimization," 2020.

[25] B. Borkowska, "Probabilistic load flow," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-93, no. 3, pp. 752–759, 1974.

[26] B. Li and J. Zhang, "A review on the integration of probabilistic solar forecasting in power systems," *Solar Energy*, vol. 210, pp. 68–86, 2020. Special Issue on Grid Integration.

[27] R. Allan, A. L. Da Silva, and R. Burchett, "Evaluation methods and accuracy in probabilistic load flow solutions," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-100, no. 5, pp. 2539–2546, 1981.

[28] M. Fan, V. Vittal, G. T. Heydt, and R. Ayyanar, "Probabilistic power flow studies for transmission systems with photovoltaic generation using cumulants," *IEEE Transactions on Power Systems*, vol. 27, no. 4, pp. 2251–2261, 2012.

[29] X. Deng, J. He, and P. Zhang, "A novel probabilistic optimal power flow method to handle large fluctuations of stochastic variables," *Energies*, vol. 10, no. 10, 2017.

[30] J. M. Morales and J. Perez-Ruiz, "Point estimate schemes to solve the probabilistic power flow," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 1594–1601, 2007.

[31] Y. Che, X. Lv, X. Wang, J. Liu, and Y. Zhang, "Probabilistic load flow using an improved point estimate method considering wind generation," in *2019 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)*, pp. 4080–4085, 2019.

[32] J. Laowanitwattana and S. Uatrongjit, "Probabilistic power flow analysis based on partial least square and arbitrary polynomial chaos expansion," *IEEE Transactions on Power Systems*, vol. 37, no. 2, pp. 1461–1470, 2022.

[33] X. Deng, P. Zhang, K. Jin, J. He, X. Wang, and Y. Wang, "Probabilistic load flow method considering large-scale wind power integration," *Journal of Modern Power Systems and Clean Energy*, vol. 7, no. 4, pp. 813–825, 2019.

[34] V. Singh, T. Moger, and D. Jena, "Probabilistic load flow approach combining cumulant method and k-means clustering to handle large fluctuations of stochastic variables," *IEEE Transactions on Industry Applications*, vol. 59, no. 3, pp. 2832–2841, 2023.

[35] M. Hajian, W. D. Rosehart, and H. Zareipour, "Probabilistic power flow by monte carlo simulation with latin supercube sampling," *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 1550–1559, 2013.

[36] X. Xu and Z. Yan, "Probabilistic load flow calculation with quasi-monte carlo and

multiple linear regression," *International Journal of Electrical Power and Energy Systems*, vol. 88, pp. 1–12, 2017.

[37] A. B. Krishna and A. R. Abhyankar, "Uniform experimental design-based nonparametric quasi-monte carlo for efficient probabilistic power flow," *IEEE Transactions on Power Systems*, vol. 38, no. 3, pp. 2318–2332, 2023.

[38] J. Yu, Y. Weng, and R. Rajagopal, "Robust mapping rule estimation for power flow analysis in distribution grids," in *2017 North American Power Symposium (NAPS)*, pp. 1–6, 2017.

[39] Y. Xu, Z. Hu, L. Mili, M. Korkali, and X. Chen, "Probabilistic power flow based on a gaussian process emulator," *IEEE Transactions on Power Systems*, vol. 35, no. 4, pp. 3278–3281, 2020.

[40] P. Pareek and H. D. Nguyen, "A framework for analytical power flow solution using gaussian process learning," *IEEE Transactions on Sustainable Energy*, vol. 13, no. 1, pp. 452–463, 2022.

[41] Y. Yang, Z. Yang, J. Yu, B. Zhang, Y. Zhang, and H. Yu, "Fast calculation of probabilistic power flow: A model-based deep learning approach," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2235–2244, 2020.

[42] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[43] X. Hu, H. Hu, S. Verma, and Z.-L. Zhang, "Physics-guided deep neural networks for power flow analysis," *IEEE Transactions on Power Systems*, vol. 36, no. 3, pp. 2082–2092, 2021.

[44] M. Yang, G. Qiu, T. Liu, J. Liu, K. Liu, and Y. Li, "Probabilistic power flow based on physics-guided graph neural networks," *Electric Power Systems Research*, vol. 235, p. 110864, 2024.

[45] A. Yaniv, P. Kumar, and Y. Beck, "Towards adoption of gnns for power flow applications in distribution systems," *Electric Power Systems Research*, vol. 216, p. 109005, 2023.

[46] A. Mohammadi and A. Kargarian, "Accelerated and robust analytical target cascading for distributed optimal power flow," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 12, pp. 7521–7531, 2020.

[47] H.-T. Zhang, W. Sun, Y. Li, D. Fu, and Y. Yuan, "A fast optimal power flow algorithm using powerball method," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 11, pp. 6993–7003, 2020.

[48] J. Lavaei and S. H. Low, "Zero duality gap in optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 92–107, 2012.

[49] A. Venzke, S. Chatzivasileiadis, and D. K. Molzahn, "Inexact convex relaxations for ac optimal power flow: Towards ac feasibility," *Electric Power Systems Research*, vol. 187, p. 106480, 2020.

[50] Z. Yuan and M. R. Hesamzadeh, "Second-order cone ac optimal power flow: convex relaxations and feasible solutions," *Journal of Modern Power Systems and Clean Energy*, vol. 7, no. 2, pp. 268–280, 2019.

[51] M. M.-U.-T. Chowdhury, S. Kamalasadan, and S. Paudyal, "A second-order cone programming (socp) based optimal power flow (opf) model with cyclic constraints for power transmission systems," *IEEE Transactions on Power Systems*, vol. 39, no. 1, pp. 1032–1043, 2024.

[52] C. Coffrin, H. L. Hijazi, and P. Van Hentenryck, "The qc relaxation: A theoretical and computational study on optimal power flow," *IEEE Transactions on Power Systems*, vol. 31, no. 4, pp. 3008–3018, 2016.

[53] M. R. Narimani, D. K. Molzahn, K. R. Davis, and M. L. Crow, "Tightening qc relaxations of ac optimal power flow through improved linear convex envelopes," *IEEE Transactions on Power Systems*, pp. 1–15, 2024.

[54] R. Louca, P. Seiler, and E. Bitar, "Nondegeneracy and inexactness of semidefinite relaxations of optimal power flow," 2014.

[55] A. Robson, M. Jamei, C. Ududec, and L. Mones, "Learning an optimally reduced formulation of OPF through meta-optimization," *CoRR*, vol. abs/1911.06784, 2019.

[56] W. Dong, Z. Xie, G. Kestor, and D. Li, "Smart-pgsim: Using neural network to accelerate ac-opf power grid simulation," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–15, 2020.

[57] D. Owerko, F. Gama, and A. Ribeiro, "Optimal power flow using graph neural networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5930–5934, 2020.

[58] 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, pp. 630–637, 2020.

[59] T. Falconer and L. Mones, "Leveraging power grid topology in machine learning assisted optimal power flow," *IEEE Transactions on Power Systems*, vol. 38, no. 3, pp. 2234–2246, 2023.

[60] M. Gao, J. Yu, Z. Yang, and J. Zhao, "A physics-guided graph convolution neural network for optimal power flow," *IEEE Transactions on Power Systems*, pp. 1–11, 2023.

[61] S. Park, W. Chen, T. W. K. Mak, and P. V. Hentenryck, "Compact optimization learning for ac optimal power flow," 2023.

[62] F. Yang, Z. Ling, Y. Zhang, X. He, Q. Ai, and R. C. Qiu, "Event detection, localization, and classification based on semi-supervised learning in power grids," *IEEE Transactions on Power Systems*, vol. 38, no. 5, pp. 4080–4094, 2023.

[63] Y. Yuan, Y. Wang, and Z. Wang, "A data-driven framework for power system event type identification via safe semi-supervised techniques," *IEEE Transactions on Power Systems*, vol. 39, no. 1, pp. 1460–1471, 2024.

[64] W. Huang and M. Chen, "Deepopf-ngt: A fast unsupervised learning approach for solving ac-opf problems without ground truth," in *ICML 2021 Workshop on Tackling Climate Change with Machine Learning*, 2021.

[65] J. Wang and P. Srikantha, "Fast optimal power flow with guarantees via an unsupervised generative model," *IEEE Transactions on Power Systems*, pp. 1–12, 2022.

[66] S. Park and P. Van Hentenryck, "Self-supervised primal-dual learning for constrained optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 4052–4060, 6 2023.

[67] M. Zhou, M. Chen, and S. H. Low, "DeepOPF-FT: One deep neural network for multiple AC-OPF problems with flexible topology," *IEEE Transactions on Power Systems*, vol. 38, no. 1, pp. 964–967, 2023.

[68] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[69] J. Yang, N. Zhang, C. Kang, and Q. Xia, "A state-independent linear power flow model with accurate estimation of voltage magnitude," *IEEE Transactions on Power Systems*, vol. 32, no. 5, pp. 3607–3617, 2017.

[70] B. Hanin, "Which neural net architectures give rise to exploding and vanishing gradients?," in *Proc. 32nd NeurIPS*, p. 580–589, Dec. 2018.

[71] 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 Transactions on Power Systems*, vol. 26, no. 1, pp. 12–19, 2011.

[72] A. B. Birchfield, T. Xu, K. M. Gegner, K. S. Shetye, and T. J. Overbye, "Grid structural characteristics as validation criteria for synthetic networks," *IEEE Transactions on Power Systems*, vol. 32, no. 4, pp. 3258–3265, 2017.

[73] C. Josz, S. Fliscounakis, J. Maeght, and P. Panciatici, "Ac power flow data in matpower and qcqp format: itesla, rte snapshots, and pegase," 2016.

[74] T. Hong, P. Pinson, and S. Fan, "Global energy forecasting competition 2012," *International Journal of Forecasting*, vol. 30, no. 2, pp. 357–363, 2014.

[75] "Solar power data for integration studies."

[76] P. P. Biswas, P. Suganthan, R. Mallipeddi, and G. A. Amaratunga, "Optimal reactive power dispatch with uncertainties in load demand and renewable energy sources adopting scenario-based approach," *Applied Soft Computing*, vol. 75, pp. 616–632, 2019.

[77] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, (Santiago, Chile), pp. 1026–1034, Dec. 2015.

[78] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[79] L. Prechelt, *Early Stopping-But When?* Berlin, Heidelberg: Springer, 2012.

[80] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, vol. 70, (Sydney, Australia), pp. 214–223, Aug. 2017.

[81] M. B. Cain, R. P. O'neill, and A. Castillo, "History of optimal power flow and formulations optimal power flow paper 1," 2012.

[82] X. Liu, Y. Dai, Y. Huang, and J. Sun, "A novel augmented lagrangian method of multipliers for optimization with general inequality constraints," *arXiv: Optimization and Control*, 2021.

[83] M. Chatzos, F. Fioretto, T. W. Mak, and P. V. Hentenryck, "High-fidelity machine learning approximations of large-scale optimal power flow," *ArXiv*, vol. abs/2006.16356, 2020.

[84] J. W. Demmel, *Applied Numerical Linear Algebra*. SIAM, 1997.

[85] R. Ribeiro, A. Street, F. Mancilla–David, and A. Angulo, "Equivalent reduced dc network models with nonlinear load functions: A data-driven approach," *IEEE Transactions on Power Systems*, vol. 39, no. 2, pp. 3021–3032, 2024.

[86] K. Chen, S. Bose, and Y. Zhang, "Unsupervised deep learning for ac optimal power

flow via lagrangian duality," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pp. 5305–5310, 2022.

[87] S. E. Kayacık, B. Kocuk, and T. Yüksel, "The promise of ev-aware multi-period optimal power flow problem: Cost and emission benefits," *Sustainable Energy, Grids and Networks*, vol. 34, p. 101062, 2023.

[88] A. R. Jha, S. Paul, and A. Dubey, "Spatially distributed multi-period optimal power flow with battery energy storage systems," in *2024 56th North American Power Symposium (NAPS)*, pp. 1–6, 2024.

[89] S. Shin, V. Rao, M. Schanen, D. A. Maldonado, and M. Anitescu, "Scalable multi-period ac optimal power flow utilizing gpus with high memory capacities," 2024.

[90] B. Akbari and G. Sansavini, "Adaptive robust ac optimal power flow considering intrahour uncertainties," *Electric Power Systems Research*, vol. 216, p. 109082, 2023.

[91] J. Liu, L. Wang, T. Jiao, W. Yang, R. Xu, K. Wu, and H. Ha, "Multi-period voltage stability-constrained optimal power flow with uncertainties," *Applied Energy*, vol. 369, p. 123522, 2024.

[92] R. Zafar and I.-Y. Chung, "Data-driven multiperiod optimal power flow for power system scheduling considering renewable energy integration," *IEEE Access*, vol. 12, pp. 95278–95290, 2024.

[93] R. Rastgoo, N. Amjady, and H. Zareipour, "A deep generative model for selecting

representative periods in renewable energy-integrated power systems," *Applied Soft Computing*, vol. 165, p. 112107, 2024.

[94] R. Hu and Q. Li, "A data-driven optimization method considering data correlations for optimal power flow under uncertainty," *IEEE Access*, vol. 11, pp. 32041–32050, 2023.

[95] B. Huang, T. Zhao, M. Yue, and J. Wang, "Bi-level adaptive storage expansion strategy for microgrids using deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 15, no. 2, pp. 1362–1375, 2024.

[96] S. Du, T. Ding, Y. Xiao, J. Wan, J. Liu, and F. Meng, "Real-time scheduling of high-penetrated renewable power systems: An expert knowledge and reinforcement learning hybrid approach," *IEEE Transactions on Power Systems*, pp. 1–13, 2024.

[97] T. Wu, A. Scaglione, and D. Arnold, "Constrained reinforcement learning for predictive control in real-time stochastic dynamic optimal power flow," *IEEE Transactions on Power Systems*, vol. 39, no. 3, pp. 5077–5090, 2024.

[98] X. Zhang, A. T. Eseye, B. Knueven, W. Liu, M. Reynolds, and W. Jones, "Curriculum-based reinforcement learning for distribution system critical load restoration," *IEEE Transactions on Power Systems*, vol. 38, no. 5, pp. 4418–4431, 2023.