Partially Permutation-Invariant Neural Network for Solving Two-Stage Stochastic AC-OPF Problem

Min Zhou, Enming Liang, Minghua Chen, *Fellow, IEEE*, and Steven H. Low, *Fellow, IEEE*

***Abstract*—We develop DeepOPF-Stoc as a neural network approach to efficiently solve the two-stage stochastic AC optimal power flow (AC-OPF) problem, which emphasizes reliable and cost-effective grid operation while accounting for uncertainties in renewable energy generation and load demand. We first identify a crucial *partial permutation-invariance* property of the mapping from load to first-stage solution. We then leverage this property in DeepOPF-Stoc to design (i) PPNN as a principled architectural framework that orchestrates standard neural network designs to learn such mappings, and (ii) a PPNN-aware sampling algorithm to prepare training data efficiently. DeepOPF-Stoc effectively addresses the dimensionality explosion issue inherent in vanilla designs, significantly reducing both the required number of neurons and the volume of training data. Our theoretical analysis confirms the universal approximation capability of PPNNs for our application. We further prove that the PPNN-aware sampling algorithm improves sampling efficiency by a factor of $K!$ compared to uniform sampling, where K represents the number of second-stage scenarios. Simulation results on IEEE 118-bus and synthetic 793-bus test systems demonstrate the superiority of DeepOPF-Stoc over state-of-the-art alternatives. It achieves two orders of magnitude speedup compared to iterative solvers while generating feasible first-stage solutions with comparable second-stage out-of-sample feasibility and 0.95% cost difference.**

***Index Terms*—Two-stage stochastic optimal power flow; neural network; deep learning; partial permutation-invariance**

NOMENCLATURE

Variable	Definition
$\mathcal{B}$	Set of buses
$\mathcal{G}$	Set of P-V buses
$\mathcal{E}$	Set of transmission lines
$\boldsymbol{g}$, $\boldsymbol{b}$	Conductance and susceptance matrix
$\overline{\boldsymbol{s}}$	Transmission line flow limit matrix
$\overline{\boldsymbol{p^g}}$, $\underline{\boldsymbol{p^g}}$	Maximum and minimum active generation vector
$\overline{\boldsymbol{q^g}}$, $\underline{\boldsymbol{q^g}}$	Maximum and minimum reactive generation vector
$\overline{\boldsymbol{v}}$, $\underline{\boldsymbol{v}}$	Maximum and minimum voltage magnitude vector
$\overline{\boldsymbol{\theta}}$, $\underline{\boldsymbol{\theta}}$	Maximum and minimum angle difference matrix
p_{ik}^d, q_{ik}^d	Active and reactive load at bus i, scenario k
p_{ijk}^f	Active power flow at line (i,j), scenario k
q_{ijk}^f	Reactive power flow at line (i,j), scenario k
v_{ik}, θ_{ik}	Voltage magnitude and angle at bus i, scenario k
p_{ik}^g, q_{ik}^g	Active and reactive generation at bus i, scenario k

This work is supported in part by a General Research Fund from Research Grants Council, Hong Kong (Project No. 11214825), a Collaborative Research Fund from Research Grants Council, Hong Kong (Project No. C1049-24G), an InnoHK initiative, The Government of the HKSAR, Laboratory for AI-Powered Financial Technologies, a Shenzhen-Hong Kong-Macau Science & Technology Project (Category C, Project No. SGDX20220530111203026), and a Start-up Research Grant from The Chinese University of Hong Kong, Shenzhen (Project No. UDF01004086), and in part by Caltech Resnick Institute of Sustainability, and S2I grant.

M. Zhou and E. Liang are with Department of Data Science, City University of Hong Kong. M. Chen is with School of Data Science, The Chinese University of Hong Kong, Shenzhen, on leave from Department of Data Science, City University of Hong Kong. S. Low is with Computing and Mathematical Sciences and Electrical Engineering, Caltech. Corresponding authors: E. Liang and M. Chen.

I. INTRODUCTION

Optimal power flow (OPF) is a fundamental problem in power system operation, aiming to serve the given load demand and minimize an objective function (e.g., generation costs) subject to physical, operational, and technical constraints by optimizing dispatch and transmission decisions [1]. OPF is critical for balancing supply and demand in real-time grid operations and has conventionally been solved using iterative schemes, such as primal-dual interior-point methods [2], and more recently, using machine learning approaches [3]–[6] (see further discussions in Section II).

In recent years, the increasing integration of renewable generation (e.g., wind and solar) and demand-side management has introduced significant uncertainties in power supply and demand. This makes it challenging for grid operators to maintain system reliability and efficiency by solving standard OPF problems, which only consider deterministic load inputs [7]. To effectively manage these uncertainties, system operators are turning to advanced uncertainty management techniques, such as multi-stage stochastic AC-OPF, to optimize dispatch and transmission decisions while explicitly accounting for future supply and demand uncertainties. Within this framework, a popular and practical scheme is the scenario-based two-stage stochastic AC-OPF, which uses a set of i.i.d. scenarios, i.e., particular realizations of renewable generation and load, from their respective distributions [8], [9], to represent uncertainties.

While effective in representing uncertainties, the scenario-based two-stage stochastic AC-OPF faces a critical challenge: the dimensionality explosion issue. This issue arises as the number of decision variables within it grows linearly with the number of collected scenarios. For practical power grids with complex uncertainties, this results in a prohibitive computational burden for conventional iterative methods [24], making them impractical for real-time grid operations. Similarly, vanilla deep neural network (DNN) with fully connected layers, while effective in solving standard OPF problems, are also vulnerable to this dimensionality explosion issue. Specifically, by concatenating the load inputs of each scenario into a high-dimensional vector and feeding it into such a vanilla fully-connected neural network (FCNN), the input dimension in these designs grows linearly with the number of scenarios,

TABLE I
EXISTING STUDIES ON MACHINE LEARNING FOR SOLVING OPF PROBLEMS.

Category	Approach	Related work	OPF model	ML model	Metrics in consideration[5]		
					Feasibility	Optimality	Speedup
Learning-aided	Predicting active constraints	[10]	DC-OPF	DNN	✓	✓	✗
	Predicting warm-start point	[3], [11]–[13]	DC-OPF	DNN	✓	✓	✗
Learning-to-optimize	Learning a solution update strategy in iterative solvers	[14]	AC-OPF	DNN	✗	✓	✓
		[15]	DC-OPF	RNN[1]	✗	✓	✓
Learning load-solution mapping	Learning the load-solution mapping and generating solutions directly from the machine learning model	[16], [17]	SC-DCOPF[2]	DNN	✓	✓	✓
		[18]	AC-OPF	DNN	✗	✓	✓
		[5], [6]	AC-OPF	DNN	✓	✓	✓
		[19]	AC-OPF	GNN[3]	✓	✓	✓
		[20]	AC-OPF	CNN[4]	✓	✓	✓
		[21], [22]	DC-OPF	DNN	✓	✓	✓
		[23]	Stochastic DC-OPF	DNN	✓	✓	✓
		This work	Stochastic AC-OPF	PPNN	✓	✓	✓

[1] RNN denotes the recurrent neural network; [2] SC-DCOPF denotes the security-constrained DC-OPF; [3] GNN denotes the graph neural network; [4] CNN denotes the convolutional neural network; [5] The column "Metrics in consideration" contains three sub-columns ("Optimality", "Feasibility", and "Speedup") that indicate whether each corresponding study addresses these performance criteria. A checkmark (✓) signifies that the method explicitly considers and evaluates the respective metric in either its solution design or performance assessment. Conversely, a cross (✗) indicates that the work does not address the corresponding metric.

making it computationally difficult for neural network training. Furthermore, this increasing input dimension demands significantly larger training datasets than those required for standard OPF cases.

To overcome these dimensionality challenges and unleash the full potential of scenario-based approaches in grid operations, we propose DeepOPF-Stoc as a novel partially permutation-invariant neural network (PPNN)-based approach to solving the scenario-based two-stage stochastic AC-OPF problem efficiently. Our scheme is based on an elegant symmetric structure of the problem: *partial permutation-invariance*. This property arises from the fact that, when solving the scenario-based two-stage stochastic AC-OPF problem, switching the order of second-stage scenario inputs does not change the first-stage solution. Our main contributions are:

▷ After reviewing the two-stage stochastic AC-OPF formulation in Section III-A, we identify its inherent partial permutation-invariance property in Section III-B.

▷ In Section IV, we exploit the partial permutation-invariance property to design PPNN as a principled architectural framework that orchestrates standard neural network (NN) designs for learning the load to first-stage solution mapping, without suffering from the dimensionality explosion issue. Theoretical analysis in Section V shows the approximation capability of PPNN for our application.

▷ Based on the PPNN design, we propose a PPNN-aware sampling algorithm to prepare the training data efficiently in Section IV-C. This algorithm improves the sampling efficiency by a factor of $K!$ compared with uniform sampling, where K denotes the number of second-stage scenarios.

▷ We conduct extensive simulations on various test cases to evaluate the performance of DeepOPF-Stoc in Section VI. The results on IEEE 118-bus and synthetic 793-bus test systems demonstrate the superiority of DeepOPF-Stoc over state-of-the-art alternatives. Our approach generates feasible solutions with comparable out-of-sample feasibility performance and achieves 0.95% out-of-sample cost difference compared to conventional iterative solvers. Notably, DeepOPF-Stoc executes two orders of magnitude faster than the state-of-the-art alternatives, marking a significant advancement in computational efficiency for solving two-stage stochastic AC-OPF problems.

II. RELATED WORK

A. Two-stage stochastic AC-OPF problem

The two-stage stochastic AC-OPF problem, which optimizes power dispatch and transmission decisions while explicitly accounting for future uncertainties in load and renewable generation, plays a crucial role in modern power system operation. Existing methods for tackling this problem include robust optimization [25]–[28], chance-constrained optimization [29]–[31], and the scenario-based approach [32].

Robust optimization aims to ensure the satisfaction of OPF constraints under all possible uncertainty realizations within a given uncertainty set, such as elliptical or polyhedral sets [26]–[28]. By optimizing the worst-case performance over all these possible scenarios, this approach typically provides a conservative decision. Instead of ensuring feasibility under all possible uncertainty realizations, chance-constrained optimization aims to find OPF solutions that satisfy constraints with a specified high probability [30], [31]. This approach offers a balance between robustness and cost-effectiveness by allowing for a small probability of constraint violation. Another widely adopted method is the scenario-based approach [33], which approximates the underlying uncertainties by sampling a set of scenarios independently from their respective probability distributions. By optimizing the OPF objective over these representative scenarios, this approach provides a practical way to solve the two-stage stochastic AC-OPF problem over general distributions, for which there are typically no closed-form expressions for the objective and constraints.

Our work focuses on the scenario-based approach for its ability to capture complex spatial and temporal correlations in renewable generation and load inputs [8], [9] while preserving the full nonlinearity of AC-OPF constraints without requiring

additional approximations. However, one key limitation of this approach is the dimensionality explosion issue, where the number of decision variables increases linearly with the number of sampled load/renewable scenarios. This growth in decision variables introduces a prohibitive computational burden for conventional iterative methods, making them impractical for real-time applications in power systems with high levels of renewable integration. We aim to address this challenge to unleash the full potential of scenario-based approaches in grid operations.

B. Machine learning for solving OPF problems

In recent years, significant efforts have been made to apply machine learning to tackle the OPF problem. As presented in Table I, existing learning-based approaches can mainly be divided into three categories: learning-aided, learning-to-optimize, and learning load-solution mapping.

Learning-aided approaches leverage machine learning as a building block to accelerate existing iterative solvers. This is achieved by identifying the active/inactive constraints of OPF problems to reduce problem size [34]–[36] or predicting a warm-start point for iterative solvers to accelerate convergence [3], [11], [37]. The learning-to-optimize approach accelerates existing iterative algorithms by introducing a learned iterative strategy that requires less iteration time [14], [15], [38]. Instead of relying on iterative optimization, the learning load-solution mapping approach solves OPF problems directly by learning a mapping from load to the optimal solution [5], [6], [16], [18], [21], [39]–[52]. A critical challenge, however, is to ensure the feasibility of the solutions obtained, given the inherent prediction error of DNNs. To tackle this challenge, the predict-and-reconstruct framework [16], [53], preventive learning [54], gauge-mapping [23], homeomorphic projection [41], and bisection projection [55] have been proposed.

The above learning-based schemes mainly focus on deterministic OPFs, and little has been done for two-stage stochastic OPF problems, which present several unique characteristics absent in deterministic formulations: (i) significantly larger input dimensions due to the consideration of numerous scenarios, (ii) substantially more complex constraint structures involving intricate coupling between first-stage and second-stage decision variables, and (iii) the inherent partial permutation-invariance property that we identify and exploit. In [32], the two-stage stochastic AC-OPF problem is recast as a min-max optimization problem and solved with an adversarial learning approach. This method, however, introduces model inaccuracy and incurs high computational complexity. In [56], [57], DNN is employed to learn a control policy for power generation considering (only) single-stage uncertainty.

In the broader context of stochastic programming, there has been growing interest in leveraging learning-based approaches to improve computational efficiency. In [58], [59], machine learning schemes are proposed to select representative scenarios when solving stochastic programming problems. However, the existence of prediction errors may result in the loss of critical scenario information, leading to an inaccurate approximation of the original stochastic programming problem. Meanwhile, several studies use DNNs as a building block to accelerate existing iterative solvers [60], [61]. However, these methods often incur a considerable computational cost due to the need for repeated gradient evaluations and iterative solution procedures. A framework for tackling two-stage stochastic problems is proposed in [62], which embeds a second-stage NN into a mixed integer problem to obtain the first-stage solution. The formulated problem, however, is computationally difficult to solve. To date, it remains largely open to solve the two-stage stochastic AC-OPF problem efficiently, i.e., significantly faster than iterative solvers, to enable its application in real-time grid operation.

In this paper, we design efficient machine learning schemes for the two-stage stochastic AC-OPF problem by exploiting its inherent partial permutation-invariance property.

III. Two-stage Stochastic AC optimal power flow

A. The two-stage stochastic AC-OPF problem

The two-stage stochastic AC-OPF problem considers a power operation over two consecutive time periods. In the first period (first stage), the load and renewable generation are given and deterministic, the operator makes decisions to meet the observed net load while allocating upward/downward reserves to accommodate uncertainties in the next period. In the second period (second stage), new load and renewable generation values are revealed. The operator then adjusts power generation to serve this newly observed net load, while respecting the ramping limits on power generation between the two periods. The objective is to minimize the sum of the first-stage generation cost and the expected second-stage one.

Our paper focuses on the popular and practical *scenario-based approach*[1] to the two-stage stochastic AC-OPF problem, which represents uncertainties by collecting a set of i.i.d. scenarios, i.e., particular realizations of renewable generation and load, sampled from their respective probability distributions. These scenarios are then used to approximate the expected second-stage cost and feasibility requirements. To this end, the scenario-based two-stage stochastic AC-OPF problem can be formulated as follows [64]:

[1]The scenario-based approach uses finite samples drawn from continuous probability distributions to approximate infinite uncertainty sets. This approach offers two key advantages: computational tractability, as it creates a manageable approximation while preserving the nonlinear AC power flow equations essential for realistic modeling; and theoretical soundness, as it guarantees convergence to the true infinite-dimensional solution as scenario count increases [63].

$$\min \ \sum_{i\in\mathcal{B}}\left[c_{i1}\cdot(p^g_{i0})^2+c_{i2}\cdot p^g_{i0}+c_{i3}\right] + \frac{1}{K}\sum_{k=1}^{K}\sum_{i\in\mathcal{B}}\left[c_{i1}\cdot(p^g_{ik})^2+c_{i2}\cdot p^g_{ik}+c_{i3}\right] \quad (1)$$

$$\text{s.t. } p^f_{ijk}=g_{ij}v^2_{ik}-v_{ik}v_{jk}(g_{ij}\cos\theta_{ijk}+b_{ij}\sin\theta_{ijk}), \ (i,j)\in\mathcal{E},\ k=0,\dots,K, \quad (2)$$

$$q^f_{ijk}=-b_{ij}v^2_{ik}-v_{ik}v_{jk}(g_{ij}\sin\theta_{ijk}-b_{ij}\cos\theta_{ijk}), \ (i,j)\in\mathcal{E},\ k=0,\dots,K, \quad (3)$$

$$p^g_{ik}-p^d_{ik}=\sum_{(i,j)\in\mathcal{E}}p^f_{ijk},\ i\in\mathcal{B},\ k=0,\dots,K, \quad (4)$$

$$q^g_{ik}-q^d_{ik}=\sum_{(i,j)\in\mathcal{E}}q^f_{ijk},\ i\in\mathcal{B},\ k=0,\dots,K, \quad (5)$$

$$\underline{p^g_i}\le p^g_{ik}\le\overline{p^g_i},\ i\in\mathcal{B},\ k=0,\dots,K, \quad (6)$$

$$\underline{q^g_i}\le q^g_{ik}\le\overline{q^g_i},\ i\in\mathcal{B},\ k=0,\dots,K, \quad (7)$$

$$\underline{v_i}\le v_{ik}\le\overline{v_i},\ i\in\mathcal{B},\ k=0,\dots,K, \quad (8)$$

$$\underline{\theta_{ij}}\le\theta_{ijk}=\theta_{ik}-\theta_{jk}\le\overline{\theta_{ij}},(i,j)\in\mathcal{E},k=0,\dots,K, \quad (9)$$

$$(p^f_{ijk})^2+(q^f_{ijk})^2\le(\overline{s_{ij}})^2,(i,j)\in\mathcal{E},\ k=0,\dots,K, \quad (10)$$

$$|p^g_{ik}-p^g_{i0}|\le\Delta p_i,\ i\in\mathcal{B},\ k=1,\dots,K, \quad (11)$$

$$\text{var. } p_{ik},q_{ik},\theta_{ik},v_{ik},\ i\in\mathcal{B},\quad k=0,\dots,K. \quad (12)$$

In this formulation, c_{i1}, c_{i2}, and c_{i3} are positive cost coefficients. K is the number of second-stage scenarios. We use $k=0$ to represent the only first-stage scenario. Constraints in (2) and (3) define the active and reactive power flow at each scenario. The power balance at each scenario is ensured by constraints (4) and (5). Constraints (6) and (7) capture the active and reactive power generation limits. The voltage magnitude and phase angle limits are described in constraints (8) and (9). Constraint (10) ensures the branch flow limit. Constraint (11) enforces ramping limits on active power generation, ensuring that the variation in active generation between consecutive time periods does not exceed the ramping limit Δp_i. The objective function in (1) minimizes the sum of the first-stage generation cost and the approximated expected second-stage generation cost. This formulation provides a comprehensive framework for addressing uncertainties while maintaining operational efficiency and reliability.

While popular and practical in addressing uncertainties, the scenario-based approach is known to suffer from the dimensionality explosion issue, as the number of decision variables grows linearly with the number of sampled scenarios. This causes a significant computational burden for conventional iterative solvers. Similarly, recent machine learning approaches, while effective in solving standard OPF problems, are also vulnerable to this dimensionality explosion issue. By concatenating the load inputs of each scenario into a high-dimensional vector and feeding it into the vanilla FCNN, the input dimension in these approaches grows linearly with the number of scenarios. This makes it computationally difficult to train the FCNN and demands a significantly larger training dataset than those required for standard OPF cases.

Next, we will identify a unique structure inherent to the scenario-based two-stage stochastic AC-OPF problem. This structure is crucial in designing machine learning schemes that avoid dimensionality explosion and employ an effective sampling method, addressing the key issues in existing approaches.

B. Partial permutation-invariance of stochastic AC-OPF

We first identify the partial permutation invariance property of the input to first-stage solution mapping in the two-stage stochastic AC-OPF problem. Subsequently, we derive an elegant expression of the first-stage solution in terms of the input, which will guide our neural network design in later sections.

Definition 1 (Permutation)**.** A *permutation* of an index set $\{1,\dots,K\}$ is a complete rearrangement of the integers 1 to K in a specific order. We use π_i to denote the i-th element of the permutation. For a vector $(\boldsymbol{x}_1,\dots,\boldsymbol{x}_K)$, we use $\pi(\boldsymbol{x}_1,\dots,\boldsymbol{x}_K)$ to denote the permuted vector $(\boldsymbol{x}_{\pi_1},\dots,\boldsymbol{x}_{\pi_K})$.

For example, a valid permutation of the index set $\{1,2,3,4\}$ could be $\pi=(3,2,1,4)$, where $\pi_1=3$, $\pi_2=2$, $\pi_3=1$ and $\pi_4=4$. For a vector $(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3,\boldsymbol{x}_4)$, the corresponding permuted vector is $\pi(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3,\boldsymbol{x}_4)=(\boldsymbol{x}_3,\boldsymbol{x}_2,\boldsymbol{x}_1,\boldsymbol{x}_4)$. This represents one of the $4!=24$ possible permutations of this four-element set.

In the two-stage stochastic AC-OPF problem, let $\boldsymbol{d}_0\in\mathbb{R}^m$ denote the first-stage net load and $\boldsymbol{d}_k\in\mathbb{R}^m$ represent the net load in the k-th second-stage scenario, the target mapping from loads to first-stage solution $\boldsymbol{u}_0\in\mathbb{R}^n$, denoted as $\mathcal{P}^*:\mathbb{R}^{m(K+1)}\to\mathbb{R}^n$, satisfying the following property:

Proposition 1. *(Partial Permutation-Invariance Property) Suppose the optimal solution of the two-stage stochastic AC-OPF problem is unique for any load input $(\boldsymbol{d}_0,\boldsymbol{d}_1,\dots,\boldsymbol{d}_K)$ in a region. Then, for any load input in the region, we must have,*

$$\mathcal{P}^*(\boldsymbol{d}_0,\boldsymbol{d}_1,\dots,\boldsymbol{d}_K)=\mathcal{P}^*\left(\boldsymbol{d}_0,\pi(\boldsymbol{d}_1,\dots,\boldsymbol{d}_K)\right), \quad (13)$$

for any permutation π of $\{1,\dots,K\}$. We say that $\mathcal{P}^*$ is partially permutation-invariant with respect to $(\boldsymbol{d}_1,\dots,\boldsymbol{d}_K)$.

Proposition 1, proved in Appendix D, formally establishes the existence and uniqueness of the target mapping $\mathcal{P}^*$ that our neural network aims to learn, contingent upon the two-stage stochastic AC-OPF problem having a unique optimal solution. Besides, it rigorously proves the partial permutation-invariance property of $\mathcal{P}^*$, specifically demonstrating invariance with respect to the ordering of second-stage load scenarios. This provides the mathematical foundation for our subsequent technical developments. Based on this property, we establish a structural decomposition of the target mapping $\mathcal{P}^*$, by applying the Kolmogorov-Arnold theorem [66] in the scenario-based two-stage stochastic AC-OPF problem setting.

Theorem 1. *For the partially permutation-invariant mapping $\mathcal{P}^*:\mathbb{R}^{m(K+1)}\to\mathbb{R}^n$, where m is the dimension of $\boldsymbol{d}_i$ and n is the dimension of $\boldsymbol{u}_0$, there exist two continuous mappings $\phi:\mathbb{R}^m\to\mathbb{R}^{b_\phi}$ and $\rho:\mathbb{R}^{m+b_\phi}\to\mathbb{R}^n$ for some $b_\phi\le K^5m^2$, such that $\mathcal{P}^*(\boldsymbol{d}_0,\boldsymbol{d}_1,\dots,\boldsymbol{d}_K)=\rho\left(\boldsymbol{d}_0,\sum_{i=1}^K\phi(\boldsymbol{d}_i)\right)$.*

Theorem 1 is proved in Appendix E. The decomposition in Theorem 1 suggests that the target mapping $\mathcal{P}^*$ can be represented by learning two separate mappings, ϕ and ρ. The theoretical bound K^5m^2 in Theorem 1 represents a conservative upper bound on the embedding dimension that may

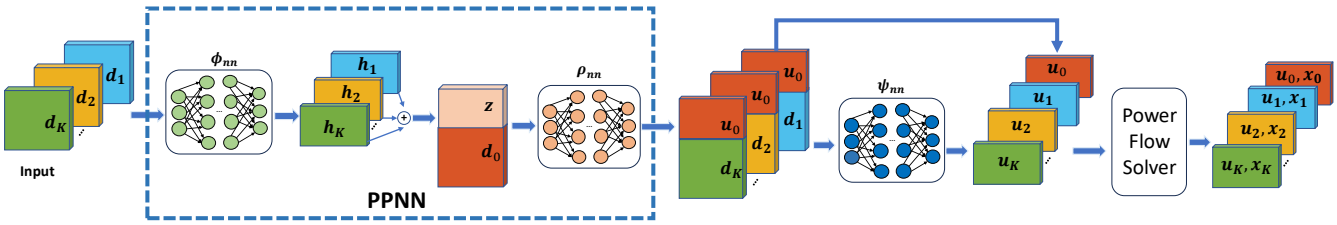


Fig. 1. Overview of DeepOPF-Stoc. First, the inputs in each second-stage scenario are fed separately into a shared neural network, denoted as ϕ_{nn}, to generate their hidden representations. Next, these representations are summed to create an embedding $\boldsymbol{z}$, which is then concatenated with the first-stage load $\boldsymbol{d}_0$ and passed to another neural network, denoted as ρ_{nn}, to generate the first-stage independent variables $\boldsymbol{u}_0$, defined in Table II. This architectural design ensures that the PPNN output remains invariant when second-stage scenarios are permuted, thereby effectively capturing and leveraging the partial permutation-invariance property inherent in two-stage stochastic AC-OPF problem. We then use a standard neural network with a power flow solver structure [5], [65] to get complete solutions of the two-stage stochastic AC-OPF problem. The $\boldsymbol{u}_0$ is concatenated with the inputs in each second-stage scenario and the first-stage scenario and passed separately to a second-stage neural network, denoted as ψ_{nn}, to generate the independent variables for each scenario. Finally, these independent variables are passed to a power flow solver to get the remaining dependent variables.

be required to learn a partially permutation-invariant mapping in the worst-case scenario. However, this bound is generally not tight for specific problem instances, as demonstrated in similar contexts within the literature (see page 16 of [67] for illustrative examples). Determining the minimal embedding dimension required for our specific target mapping $\mathcal{P}^*$ remains an open theoretical challenge.

In our practical implementation, we adopt an empirical approach that is standard in the field: we incrementally increase the embedding dimension until achieving satisfactory performance metrics, such as low cost differences and high out-of-sample feasibility rates [67]. Through extensive experimentation, we observe that setting the embedding dimension comparable to the per-scenario load input dimension m, independent of the number of scenarios K, suffices to achieve excellent performance.

Discussion: The two-stage stochastic AC-OPF problem is non-convex, potentially yielding multiple optimal solutions [49]. In this case, through the augmented learning framework [49], it is straightforward to show that the augmented mapping from the initial condition (starting point utilized in the iterative solver) and load to the first-stage solution is single-valued and partially permutation-invariant. Consequently, our approach proposed in this paper can be applied to learn this single-valued augmented mapping.

IV. DeepOPF-Stoc

In this section, we first provide an overview of our DeepOPF-Stoc design. We then introduce the PPNN architecture in detail. Following this, we present the PPNN-aware sampling algorithm to efficiently prepare the training data. Finally, we describe the training process for DeepOPF-Stoc.

A. Overview

Following the structural representation in Section III-B, we propose DeepOPF-Stoc as a PPNN-based approach to solve the two-stage stochastic AC-OPF problem efficiently. Figure 1 illustrates the DeepOPF-Stoc architecture. Unlike vanilla DNN designs that concatenate the load inputs in each scenario into a single vector and feed it into an FCNN, DeepOPF-Stoc feeds the inputs in each scenario separately into the PPNN model to generate a set of independent variables of the first-stage solution. These independent variables are then concatenated with the inputs in each second-stage scenario and passed separately to a second-stage NN to generate the independent variables for each scenario. Finally, the independent variables are passed to a power flow solver to get the remaining dependent variables.

TABLE II
SELECTED INDEPENDENT AND DEPENDENT VARIABLES FOR SCENARIO k.

	Slack bus	**P-Q bus**	**P-V bus**
$\boldsymbol{u}_k$	θ_{0k}, v_{0k}	$p^d_{ik}, q^d_{ik}, i \in \mathcal{B}$	$p^g_{ik}, v_{ik}, i \in \mathcal{G}$
$\boldsymbol{x}_k$	p^g_{0k}, q^g_{0k}	$\theta_{ik}, v_{ik}, i \in \mathcal{B}$	$\theta_{ik}, q^g_{ik}, i \in \mathcal{G}$

B. The DeepOPF-Stoc architecture

We exploit the partial permutation-invariance property of the structural representation in Theorem 1 to design PPNN as a principled architectural framework that orchestrates standard NN designs to learn the target mapping $\mathcal{P}^*$ effectively, with each component serving a specific purpose. The PPNN architecture is shown in Figure 1. Instead of concatenating the load inputs into a high-dimensional vector, PPNN first uses a neural network, denoted as ϕ_{nn}, to process the inputs of each second-stage scenario separately:

$$\boldsymbol{h}_k = \phi_{\text{nn}}(\boldsymbol{d}_k), \quad k = 1, \ldots, K, \tag{14}$$

where $\boldsymbol{d}_k$ is the load inputs of the k-th scenario. ϕ_{nn} is an FCNN with ReLU activation functions to approximate the continuous mapping ϕ in Theorem 1. Then, the hidden representations $(\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_K)$ are aggregated through summation to get the embedding $\boldsymbol{z} = \sum_{k=1}^{K} \boldsymbol{h}_k$. Then, we feed the first-stage input $\boldsymbol{d}_0$ and the embedding $\boldsymbol{z}$ into another neural network, denoted by ρ_{nn}, to obtain $\boldsymbol{u}_0$, the set of independent variables in the first-stage solution:[2]

$$\boldsymbol{u}_0 = \rho_{\text{nn}}(\boldsymbol{d}_0, \boldsymbol{z}) = \rho_{\text{nn}}\left(\boldsymbol{d}_0, \sum_{k=1}^{K} \boldsymbol{h}_k\right), \tag{15}$$

where ρ_{nn} is also an FCNN with ReLU activation functions to approximate the continuous mapping ρ in Theorem 1.

[2]For ease of discussions, we define $\boldsymbol{u}_0$ as in Table II, including both the independent variables in the first-stage solution and the first-stage load input.

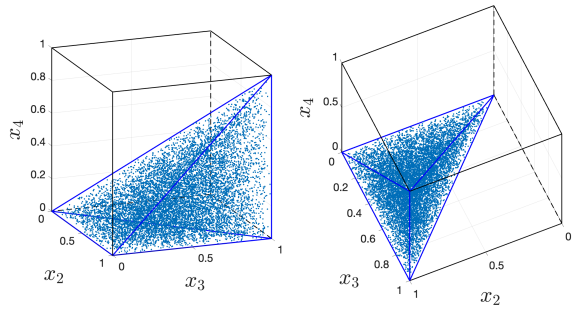


Fig. 2. A three-dimensional visualization of the samples obtained by the PPNN-aware sampling algorithm. The target mapping is $\mathcal{P}(\boldsymbol{x}) = x_1 + \frac{1}{2}(x_2 + x_3 + x_4)$, with the input domain set as $[0, 1]^4$. To eliminate sample redundancy, we employ the PPNN-aware sampling algorithm to sample from the ordered space defined by $x_2 > x_3 > x_4$. Consequently, the volume of the sampling space is reduced by a factor of 3!.

Similar to [62], to get the second-stage solutions, we train one DNN to solve multiple second-stage sub-problems by learning a mapping $\psi_{\text{nn}}(\cdot)$ from any first-stage solution $\boldsymbol{u}_0$ and second-stage input $\boldsymbol{d}_k$ to the second-stage solution,

$$\boldsymbol{u}_k = \psi_{\text{nn}}\left(\boldsymbol{u}_0, \boldsymbol{d}_k\right), \ k = 1, \ldots, K. \tag{16}$$

Finally, we solve the remaining dependent variables for each scenario, as listed in Table II, using a power flow solver, denoted as $\text{PF}(\cdot)$:

$$\boldsymbol{x}_k = \text{PF}\left(\boldsymbol{u}_k\right), \quad k = 0, \ldots, K. \tag{17}$$

In our DeepOPF-Stoc design, we set the input and output dimensions of ρ_{nn}, ϕ_{nn} and ψ_{nn} to be $O(m)$, independent of the number of second-stage scenarios. This effectively mitigates the dimensionality explosion issue encountered in vanilla DNN designs, leading to significantly reduced computational costs during both training and inference.

Discussion: The PPNN architectural framework is inspired by the function decomposition in Theorem 1, which mathematically demonstrates that the target mapping $\mathcal{P}^*$ can be uniquely decomposed into two interconnected sub-mappings with well-defined structural properties. Each component of the PPNN directly implements this theoretical decomposition and serves an integral role in learning the target mapping accurately. This makes component-wise ablation studies inapplicable, as removing individual components would violate PPNN's specific theoretical foundation for achieving partial permutation invariance. Meanwhile, while alternative architectures, such as sufficiently large FCNNs, can also achieve partial permutation-invariance through different approaches, they typically face dimensionality explosion issues as input dimensions grow linearly with the number of second-stage scenarios. PPNN, on the other hand, provides an efficient and principled approach for capturing the partial permutation-invariance property.

C. The PPNN-aware sampling algorithm

To prepare data for training the PPNN, we design an efficient PPNN-aware sampling algorithm that exploits the permutation-invariance property of the structural representation in Theorem 1. Unlike widely used uniform/Gaussian sampling strategies, which can inefficiently generate multiple different samples that are essentially equivalent due to permutation invariance, our approach avoids this redundancy.

For example, using uniform sampling, we may generate samples $\boldsymbol{l}_1 \in \mathcal{N}(\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_K)$ and $\boldsymbol{l}_2 \in \mathcal{N}(\boldsymbol{d}_0, \pi\left(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K\right))$, where π is a permutation of $\{1, \ldots, K\}$ and $\mathcal{N}(\boldsymbol{x})$ represents a neighborhood around $\boldsymbol{x}$ with nearly identical first-stage solutions. $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$, while different in their raw representation, map to nearly identical outputs, with $\mathcal{P}^*(\boldsymbol{l}_1) \approx \mathcal{P}^*(\boldsymbol{l}_2)$. This creates redundancy in the training data, leading to inefficient use of computational resources, as the PPNN is repeatedly exposed to essentially the same information in different forms. Furthermore, the PPNN is learning multiple representations of the same underlying relationship, which is unnecessary and potentially detrimental to generalization.

Based on the above observation, we propose a novel PPNN-aware sampling algorithm to prepare the training data efficiently. Instead of sampling uniformly at random from the entire input domain, our algorithm targets a representative subspace of it to exclude redundant permutations. This is achieved by sampling $(\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_K)$ from the following joint distribution:

$$f_{\boldsymbol{D}_0,\ldots,\boldsymbol{D}_K}\left(\boldsymbol{d}_0, \ldots, \boldsymbol{d}_K\right) = \begin{cases} K! \prod_{i=0}^{K} f_{\boldsymbol{D}_i}(\boldsymbol{d}_i), & \text{if } \boldsymbol{d}_1^1 > \ldots > \boldsymbol{d}_K^1, \\ 0, & \text{otherwise.} \end{cases}$$

Here, $f_{\boldsymbol{D}_0,\ldots,\boldsymbol{D}_K}\left(\boldsymbol{d}_0, \ldots, \boldsymbol{d}_K\right)$ is the joint distribution of $(\boldsymbol{d}_0, \ldots, \boldsymbol{d}_K)$ and $f_{\boldsymbol{D}_i}(\boldsymbol{d}_i)$ is the probability distribution of vector $\boldsymbol{d}_i$, with $\boldsymbol{d}_i^1$ denoting its first element. By sampling from the joint distribution $f_{\boldsymbol{D}_0,\ldots,\boldsymbol{D}_K}\left(\boldsymbol{d}_0, \ldots, \boldsymbol{d}_K\right)$[3], we focus on samples that satisfy the order requirement. All the permuted variants of the obtained samples will violate this order requirement and be excluded from the sampling process. In this way, we eliminate the redundant samples when preparing the training data. As the number of possible permutations for an input is $K!$, we can improve the sampling efficiency by this factor. See Figure 2 as an illustration of the obtained samples.

D. DeepOPF-Stoc training

Based on the training data obtained using the PPNN-aware sampling algorithm, we train DeepOPF-Stoc in a supervised manner. The loss function is designed as follows:

$$\mathcal{L} = w_1 \mathcal{L}_{\text{cost}} + w_2 \mathcal{L}_{\text{pen}} + w_3 \mathcal{L}_{\text{pred}}, \tag{18}$$

where $\mathcal{L}_{\text{pred}}$ represents the mean squared error between the predicted solution and the ground truth in the training data. $\mathcal{L}_{\text{pen}}$ is the violation of constraints in (6)-(11). $\mathcal{L}_{\text{cost}}$ represents the cost difference, calculated as the mean absolute error between the generation cost of the generated solutions and the ground truth. w_1, w_2, and w_3 are three empirically decided coefficients. By minimizing this loss function, we aim to guide the PPNN toward generating feasible solutions that closely approximate the optimal solutions.

We use the standard Adam optimizer to train DeepOPF-Stoc [68]. The gradient of $\mathcal{L}$ with respect to the parameters of ψ_{nn}, denoted as θ_ψ, is given by:

[3]We use the standard Markov Chain Monte Carlo approach to sample from the joint distribution $f_{\boldsymbol{D}_0,\ldots,\boldsymbol{D}_K}\left(\boldsymbol{d}_0, \ldots, \boldsymbol{d}_K\right)$.

$$\nabla_{\theta_\psi}\mathcal{L} = \sum_{k=1}^{K}\left[\left(\frac{\partial\mathcal{L}}{\partial \boldsymbol{u}_k} + \frac{\partial\mathcal{L}}{\partial \boldsymbol{x}_k}\frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{u}_k}\right)\frac{\partial \boldsymbol{u}_k}{\partial \theta_\psi}\right].$$

Here, $\partial\mathcal{L}/\partial\boldsymbol{u}_k$ and $\partial\mathcal{L}/\partial\boldsymbol{x}_k$ can be directly calculated based on the explicit form of the loss function in (18). $\partial\boldsymbol{u}_k/\partial\theta_\psi$ is the gradient of ψ_{nn}'s output $\boldsymbol{u}_k$ to its parameters θ_ψ, which can be calculated using standard backpropagation based on (16). $\partial\boldsymbol{x}_k/\partial\boldsymbol{u}_k$ is calculated using the implicit gradient approach in [5]. Next, we calculate the gradient of $\mathcal{L}$ to the parameters of ρ_{nn}, denoted as $\theta\rho$, as:

$$\nabla_{\theta_\rho}\mathcal{L} = \sum_{k=0}^{K}\left[\left(\frac{\partial\mathcal{L}}{\partial \boldsymbol{u}_k} + \frac{\partial\mathcal{L}}{\partial \boldsymbol{x}_k}\frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{u}_k}\right)\frac{\partial \boldsymbol{u}_k}{\partial \boldsymbol{u}_0}\frac{\partial \boldsymbol{u}_0}{\partial \theta_\rho}\right].$$

Here, $\partial\boldsymbol{u}_k/\partial\boldsymbol{u}_0$ is the gradient of neural network ψ_{nn}'s output $\boldsymbol{u}_k$ to its input $\boldsymbol{u}_0$, and $\partial\boldsymbol{u}_0/\partial\theta_\rho$ is the gradient of neural network ρ_{nn}'s output $\boldsymbol{u}_0$ to its parameters θ_ρ. Based on (16) and (15), both gradients can be calculated using the standard backpropagation. Finally, we compute the gradient of $\mathcal{L}$ to the parameters of ϕ_{nn}, denoted as θ_ϕ:

$$\nabla_{\theta_\phi}\mathcal{L} = \sum_{k=0}^{K}\left[\left(\frac{\partial\mathcal{L}}{\partial \boldsymbol{u}_k} + \frac{\partial\mathcal{L}}{\partial \boldsymbol{x}_k}\frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{u}_k}\right)\frac{\partial \boldsymbol{u}_k}{\partial \boldsymbol{u}_0}\frac{\partial \boldsymbol{u}_0}{\partial \boldsymbol{z}}\left(\sum_{i=1}^{K}\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{h}_i}\frac{\partial \boldsymbol{h}_i}{\partial \theta_\phi}\right)\right].$$

Here, $\partial\boldsymbol{u}_0/\partial\boldsymbol{z}$ is the gradient of neural network ρ_{nn}'s output $\boldsymbol{u}_0$ with respect to its input $\boldsymbol{z}$, $\partial\boldsymbol{z}/\partial\boldsymbol{h}_i$ equals to 1 as $\boldsymbol{z} = \sum_{k=1}^{K}\boldsymbol{h}_k$, and $\partial\boldsymbol{h}_i/\partial\theta_\phi$ is the gradient of neural network ϕ_{nn}'s output $\boldsymbol{z}$ with respect to its parameters θ_ϕ. Based on (15) and (14), we can use standard backpropagation techniques to compute both $\partial\boldsymbol{u}_0/\partial\boldsymbol{z}$ and $\partial\boldsymbol{h}_i/\partial\theta_\phi$.

V. PPNN Performance Analysis

In this section, we analyze the capability of PPNN to approximate the load to first-stage solution mapping $\mathcal{P}^*$ of the two-stage stochastic AC-OPF problem. The results justify and also guide the design of ρ_{nn} and ϕ_{nn} in DeepOPF-Stoc.

Definition 2. Let $\boldsymbol{l} := (\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_K) \in \mathcal{D}$ denote the load input, where each $\boldsymbol{d}_k \in \mathbb{R}^m$ and $\mathcal{D}$ represents the load domain. We define the error of approximating $\mathcal{P}^* : \mathbb{R}^{m(K+1)} \to \mathbb{R}^n$ by a composite mapping consisting of continuous mappings $\phi : \mathbb{R}^m \to \mathbb{R}^{b_\phi}$ and $\rho : \mathbb{R}^{m+b_\phi} \to \mathbb{R}^n$ as

$$\epsilon(b_\phi) \triangleq \inf_{\substack{\phi\in\mathcal{F}\\ \rho\in\mathcal{Q}}}\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\mathcal{P}^*(\boldsymbol{l}) - \rho\left(\boldsymbol{d}_0, \sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2, \tag{19}$$

where b_ϕ is a given embedding dimension. $\mathcal{F}$ is the set of all continuous mappings from $\mathbb{R}^m$ to $\mathbb{R}^{b_\phi}$ and $\mathcal{Q}$ denotes the set of all continuous mapping from $\mathbb{R}^{m+b_\phi}$ to $\mathbb{R}^n$.

Note that according to Theorem 1 in Section III-B, the error $\epsilon(b_\phi)$ in (19) is zero when $b_\phi \geq K^5m^2$. For smaller b_ϕ, however, the approximation error may be positive.

The following theorem states that there exists a PPNN that can approximate $\mathcal{P}^*$ well.

Theorem 2. *For any continuous mappings ρ^* and ϕ^* that achieve the minimal $\epsilon(b_\phi)$ in* (19)*, there exists a PPNN composed of neural networks ρ_{nn} and ϕ_{nn}, such that its approximation error to $\mathcal{P}^*$, defined as*

$$\epsilon_p \triangleq \inf_{\substack{\phi\in\mathcal{H}\\ \rho\in\mathcal{V}}}\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\mathcal{P}^*(\boldsymbol{l}) - \rho\left(\boldsymbol{d}_0, \sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2,$$

is bounded by

$$\epsilon(b_\phi) \leq \epsilon_p \leq \epsilon(b_\phi) + K\,\mathrm{Lip}(\rho^*)\epsilon_{\phi_{\text{nn}}} + \epsilon_{\rho_{\text{nn}}},$$

where $\mathcal{H}$ and $\mathcal{V}$ are the set of functions represented by ϕ_{nn} and ρ_{nn}, respectively. $\mathrm{Lip}(\rho^*)$ denotes the Lipschitz constant of ρ^*, and $\epsilon_{\phi_{\text{nn}}}$ and $\epsilon_{\rho_{\text{nn}}}$ are given by

$$\epsilon_{\phi_{\text{nn}}} \triangleq \inf_{\phi\in\mathcal{H}}\sup_{\boldsymbol{x}\in\mathcal{X}}\left\|\phi^*(\boldsymbol{x}) - \phi(\boldsymbol{x})\right\|_2,$$

$$\epsilon_{\rho_{\text{nn}}} \triangleq \inf_{\rho\in\mathcal{V}}\sup_{\boldsymbol{y}\in\mathcal{Y}}\left\|\rho^*(\boldsymbol{y}) - \rho(\boldsymbol{y})\right\|_2,$$

where $\mathcal{X} = \mathbb{R}^m$ and $\mathcal{Y} = \mathbb{R}^{m+b_\phi}$ are the domains of ϕ^* and ρ^*, respectively.

Theorem 2, proved in Appendix F, bounds the approximation error of PPNN to the target mapping $\mathcal{P}^*$. It provides strong justification for choosing the particular PPNN architecture to learn the target mapping $\mathcal{P}^*$.

First, it establishes the fundamental approximation capability of our PPNN architecture. With sufficient parameters in ρ_{nn} and ϕ_{nn}, our PPNN can theoretically approximate the target mapping $\mathcal{P}^*$ to within the fundamental error lower bound $\epsilon(b_\phi)$. This serves a similar purpose to the classical universal approximation theorem for fully connected neural networks, providing practitioners with confidence that the PPNN architecture is fundamentally capable of learning the target mapping, rather than being limited by structural constraints.

Second, the theorem demonstrates that the overall PPNN approximation error ϵ_{p} scales linearly with the number of second-stage scenarios K, contrasting favorably with the exponential growth typical in standard FCNNs. This theoretical scaling advantage provides important insights into why PPNNs should be preferred for this class of problems. Our simulation results in Section VI-C, particularly Figure 4 and the accompanying discussions, corroborate this theoretical advantage and help explain the superior practical performance we observe.

VI. Numerical Experiments

A. Experimental setup

We evaluated the performance of DeepOPF-Stoc on modified IEEE 118-bus and synthetic 793-bus test systems from the Power Grid Library [69]. To simulate renewable generation, we modify the test system by adding negative load demand in selected buses (see our test cases in [70]). All simulations were conducted in a CentOS 7.6 environment with a quad-core (E5-2609@2.50GHz) CPU and 59GB RAM. We use the standard MIPS solver to solve the scenario-based two-stage stochastic AC-OPF problem and use the obtained solutions as ground truths.[4] For each test case, we create 10,000 instances

[4]We note that the scenario-based formulation transforms the original two-stage stochastic AC-OPF problem into a deterministic one, which is solvable using the MIPS solver.

for training and 2,500 for testing. We implement DeepOPF-Stoc based on the PyTorch platform, with hyperparameters summarized in Table III. We use the following feasibility-recovery mechanism (FRM) to ensure solution feasibility if a feasible solution can be successfully identified: (i) apply NN-based bisection projection to recover feasibility when possible (see [55]), and (ii) employ the obtained solution as a warm-start point for MIPS and explore alternative initial conditions if convergence fails when NN cannot identify a feasible solution.

We introduce six scenario generation schemes: three based on Gaussian distributions (with low, medium, and high levels of uncertainty) and three based on Uniform-Laplacian distributions (also with low, medium, and high uncertainty levels) to evaluate the performance of DeepOPF-Stoc over a broad range of uncertainty distributions and levels in real-world grid operations.

Gaussian Distribution: (i) **first-stage:** Load and renewable generation are sampled uniformly at random within [80%, 120%] of their default values associated with their test cases. (ii) **second-stage:** Load and renewable generation follow Gaussian distributions,[5] modeling the aggregated load consumption of numerous consumers and distributed renewable generation. The mean of the Gaussian distribution is sampled uniformly at random within [80%, 120%] of the nominal values associated with their test cases. Informed by existing studies [29], [33], we vary the standard deviation as 5%, 10%, and 15% of the corresponding mean to represent low, medium, and high uncertainty levels.

Uniform-Laplacian Distribution: (i) **first-stage:** Load and renewable generation are sampled uniformly at random within [80%, 120%] of their default values associated with their test cases. (ii) **second-stage:** The load follows a uniform distribution with the mean sampled uniformly at random within [80%, 120%] of the nominal values associated with their test cases, modeling short-term load fluctuations with equal probability. The lower and upper bounds of the distribution are set to 80% and 120% of the corresponding mean, respectively. The renewable generation is sampled from the Laplacian distribution, modeling renewable generation with a high probability of extreme fluctuations. The location parameters of the Laplacian distribution are sampled uniformly at random within [80%, 120%] of the default values. Similar to the Gaussian distribution, we vary the scale parameter as 10%, 15%, and 20% of the corresponding location parameter to represent low, medium, and high uncertainty levels.

In the test process, we examine the performance of the proposed approach on distributions from the same parametric families but with parameter values that differ from those encountered during training, which represents realistic operational conditions where distribution parameters may shift over time.

[5]We employ non-truncated Gaussian distributions instead of truncated ones for two key reasons. First, non-truncated Gaussian distributions are extensively established in power systems literature as the standard modeling approach for load demand and renewable generation uncertainties, ensuring our evaluation reflects realistic operational conditions. Second, including tail events provides a rigorous stress test of our method's resilience under challenging scenarios that may occur in real-world operations. Truncating these distributions could artificially remove extreme but realistic events, potentially leading to overly optimistic performance assessments that may not reflect true operational robustness.

TABLE III
DNN PARAMETER SETTINGS.

# Bus	# hidden layers	# hidden neurons	Batch size	Training epoch	Learning rate	w_1	w_2	w_3	b_ϕ
118	8	128	32	20,000	1e-3	0.3	0.5	1	10
793	8	512	16	5,000	1e-3	0.3	2	1	100

Baseline methods: To provide performance benchmarks, we compare DeepOPF-Stoc against the following baseline approaches: (i) the standard MIPS solver for the scenario-based two-stage stochastic AC-OPF problem [2], which serves as a high-quality local benchmark; (ii) the chance-constrained AC-OPF approach (introduced in detail in Appendix H);[6] (iii) the deterministic model, which simplifies the scenario-based approach by considering the future uncertainty as a deterministic vector. All baseline methods are implemented in Python.

Evaluation metrics: Following standard practice for scenario-based approaches [23], [71], we evaluate the performance of each method using an out-of-sample analysis. For each first-stage solution obtained, we randomly generate $K = 1000$ out-of-sample scenarios from the in-sample distribution. For each first-stage solution and second-stage scenario, the second-stage sub-problem (Appendix G) is solved using a standard MIPS solver [2], one of the state-of-the-art solvers for solving non-convex optimization problems. We consider the following evaluation metrics: (i) **Out-of-sample feasibility rate (η_{oos})**: the proportion of out-of-sample scenarios for which a feasible second-stage solution exists.[7] (ii) **Out-of-sample cost (C_{oos})**: the sum of the first-stage generation cost and the average second-stage cost over feasible out-of-sample scenarios (scenarios that lead to infeasibility in the second-stage subproblem are excluded from the averaging). (iii) **Runtime (t)**: the runtime for solving the two-stage stochastic AC-OPF problem. (iv) **Value of the stochastic solution (VSS):** The benefit of the method over the deterministic baselines. See Appendix A for detailed mathematical definitions of the evaluation metrics.

B. Performance across varying uncertainty

We begin our analysis with the modified IEEE 118-bus systems to show of the performance of DeepOPF-Stoc under various uncertainty levels and distributions. The sampling process of load and renewable generation are described in Section VI-A. The number of in-sample scenarios is set to 25. This choice of in-sample sizes reflects a balance between uncertainty representation and computational tractability when preparing the training data.

[6]We selected chance-constrained optimization instead of the robust optimization as our baseline for several methodological advantages: it generalizes robust optimization by relaxing constraint satisfaction probability from unity to $1-\delta$ (where δ represents operator-specified risk tolerance), accommodates both bounded and unbounded uncertainty sets more naturally, and enjoys broader adoption within the power systems research community.

[7]The out-of-sample feasibility rate and out-of-sample cost are two complementary metrics to evaluate the cost efficiency and operational reliability of the obtained solutions.

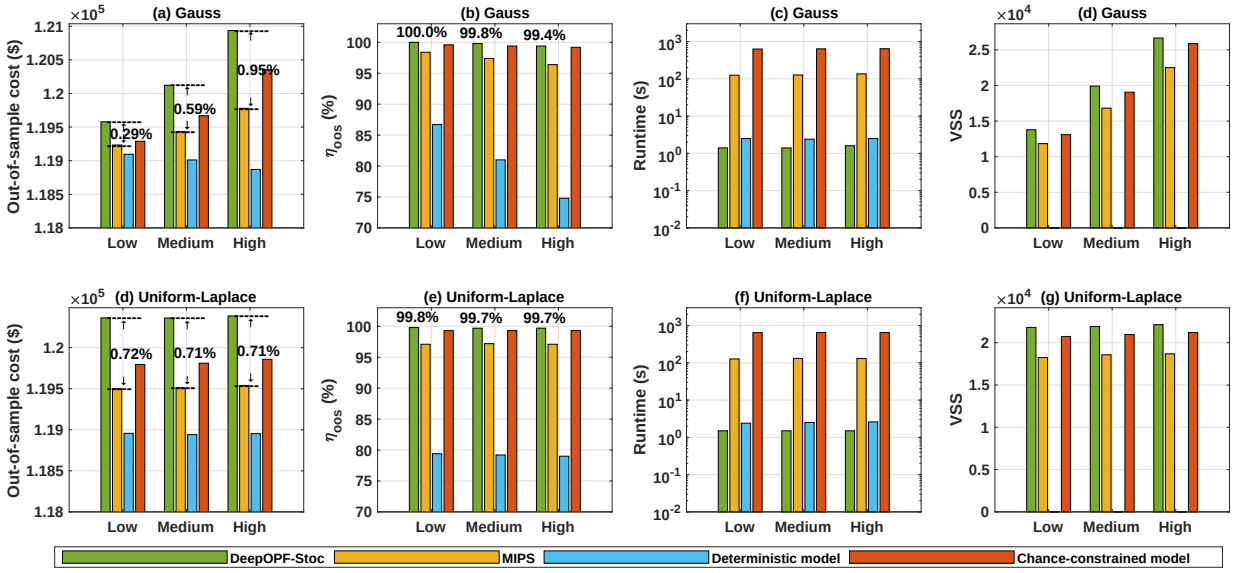


Fig. 3. Performance comparison of various methods under diverse uncertainty levels and distributions on the modified IEEE 118-bus test system.

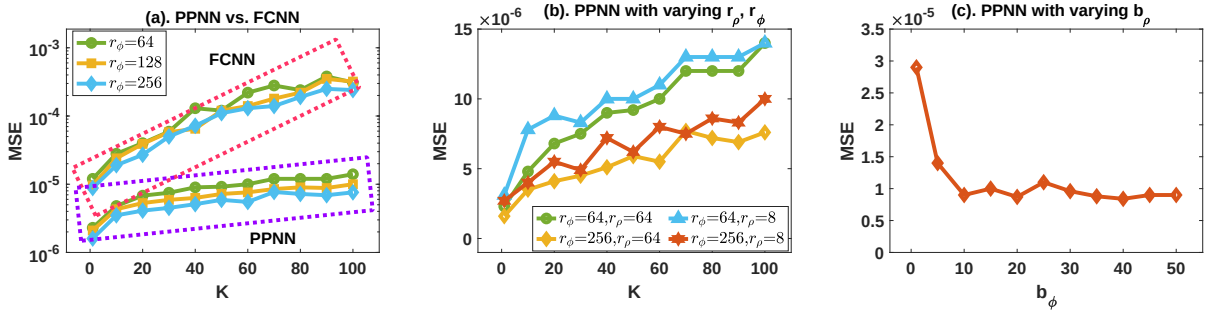


Fig. 4. MSE comparison of different methods across varying in-sample/network sizes.

Figure 3 shows that DeepOPF-Stoc consistently generates feasible first-stage solutions across various uncertainty levels and distributions, achieving out-of-sample feasibility rates comparable to MIPS while maintaining a cost difference within 0.95%.[8] Furthermore, DeepOPF-Stoc achieves a remarkable two-order-of-magnitude speedup over MIPS and a three-order-of-magnitude speedup over the chance-constrained AC-OPF approach. As compared to the deterministic model approach, DeepOPF-Stoc achieves a significantly higher out-of-sample feasibility rate, a slightly higher out-of-sample cost, and a high VSS, while being one order of magnitude faster.

We also observed that the deterministic model yields the lowest out-of-sample feasibility among the four methods. This highlights the significance of considering uncertainties in the optimization process when managing the increasing variability of load demand and renewable generation. Additionally, with the increasing level of uncertainties, all four methods exhibit either lower out-of-sample feasibilities or higher out-of-sample costs, as higher uncertainty levels either require increased reserves or lead to lower out-of-sample feasibility rates.

[8]We note that this cost difference is controllable and systematically manageable. The cost difference is fundamentally determined by the approximation capability of our PPNN model. Given sufficient training data, users can achieve an arbitrarily small cost difference by enhancing the PPNN's approximation capacity through increased network parameters and architectural sophistication.

C. PPNN approximation error

We conduct simulations on the modified IEEE 118-bus test system to analyze the approximation error of PPNN to our target mapping $\mathcal{P}^*$ by varying the number of in-sample scenarios K from 1 to 100 and setting the architectures of ϕ_{nn} and ρ_{nn} as $[236, 64, r_\phi, 64, 236]$ and $[472, 64, r_\rho, 64, 107]$, where r_ρ and r_ϕ denote the number of hidden neurons in the corresponding hidden layer. We train the PPNN model by minimizing the mean squared error (MSE) between its output and the ground truth and take the MSE over the training dataset as the empirical approximation error. We ensure convergence by extending training to 100,000 epochs. The uncertainties in load and renewable generation are modeled using the Gaussian distribution with the high uncertainty level, in which the variance of the Gaussian distribution is 15% of the corresponding mean, described in Section VI-A.

We first compare the approximation capability of PPNN against vanilla DNN designs. The latter concatenates the first-stage and second-stage inputs as a high-dimensional vector and feeds it into an FCNN. Using $r_\rho = 64$ and varying $r_\phi \in \{64, 128, 256\}$, we evaluate MSE of PPNN across different ϕ_{nn} sizes. For each PPNN configuration, the FCNN is designed with a comparable number of tunable parameters. To isolate the architectural effects, we implement experiments using (i) a standard random sampling strategy (sampling

randomly from Gaussian distributions) for FCNN, and (ii) our proposed PPNN-aware sampling strategy for PPNN, while maintaining the same total number of training samples for both models. As shown in Figure 4. (a), the approximation error of FCNN is two orders of magnitude larger than that of PPNN, highlighting the advantage of our DeepOPF-Stoc design in learning the target mapping $\mathcal{P}^*$.

We then investigate the influence of ρ_{nn} size on the approximation error of PPNN by setting $r_\rho \in \{8, 64\}$ and $r_\phi \in \{64, 256\}$. As shown in Figure 4. (b), the MSE of PPNN increases approximately linearly with the K. For a fixed r_ϕ, the slope of the MSE with respect to K remains consistent, while the entire curve shifts upward when r_ρ decreases. This behavior aligns with our theoretical analysis in Theorem 2, where the approximation errors of ϕ_{nn} and ρ_{nn} determine the slope and vertical position of the curve,[9] respectively.

Finally, we investigate the influence of the hidden dimension b_ϕ on the PPNN approximation error by fixing r_ϕ and r_ρ at 64, and varying b_ϕ from 1 to 50. Figure 4. (c) illustrates that the PPNN approximation error decreases rapidly with increasing size of b_ϕ and a relatively small hidden dimension (e.g., $b_\phi = 10$) is sufficient to achieve a high accuracy. This shows the effectiveness of our DeepOPF-Stoc design in addressing the dimensionality explosion issue of the scenario-based two-stage stochastic AC-OPF problem.

D. Scalability of DeepOPF-Stoc

We conduct simulations on a large 793-bus test system to evaluate the scalability of DeepOPF-Stoc. The number of in-sample scenarios is set to 5, 10, or 15. We model uncertainty in load and renewable generation using the Gaussian distribution with a medium uncertainty level scheme described in Section VI-A.

Simulation results in Table IV show that MIPS requires over 2,000 seconds on average to solve the two-stage stochastic AC-OPF problem for the 793-bus test system with 15 scenarios. Such runtimes are slow for real-time operations in grids with a high penetration of renewable energy, where control actions must be updated frequently, e.g., every few minutes, to respond to rapid generation changes [72]. In contrast, our proposed method is capable of solving the same problem with up to two orders of magnitude speedup, with a cost difference within 0.52% and a high out-of-sample feasibility rate. This marks a significant advancement toward real-time decision-making for power system operation under uncertainty.

Among the four methods, the deterministic model exhibits the lowest out-of-sample feasibility rate among the compared methods, highlighting the importance of incorporating uncertainty into the problem formulation. As compared to the chance-constrained AC-OPF approach, DeepOPF-Stoc achieves comparable out-of-sample costs and out-of-sample feasibility rates while being up to three orders of magnitude faster. Furthermore, the speedup achieved by DeepOPF-Stoc increases with the number of in-sample scenarios, further demonstrating its scalability.

We also observe that the out-of-sample feasibility rate of MIPS improves as the number of in-sample scenarios increases, which is consistent with the stochastic programming theory. While further increasing the in-sample size would likely yield additional feasibility improvements, our current computational resource constraints limit our ability to generate the extensive labeled training datasets required for such experiments. Our present experimental design strikes a balance between computational tractability and effective uncertainty representation by selecting scenario counts that achieve high out-of-sample feasibility rates while maintaining reasonable computational requirements.

[9]Our theoretical analysis employs the maximal error to measure the approximation error of PPNN. In the simulation, we observed similar behaviors for the mean squared error in response to variations in K and the approximation errors of ϕ_{nn} and ρ_{nn}.

VII. Conclusion and future direction

We propose DeepOPF-Stoc as the *first* neural-network approach to solve the essential two-stage stochastic AC-OPF problem efficiently. We first identify the partial permutation-invariance property of the mapping from load to the first-stage solution. Based on this property, we (i) develop a PPNN-based approach to learn such mappings without suffering from the dimensionality explosion issue in vanilla designs, and (ii) design a PPNN-aware sampling algorithm to prepare the training data efficiently. Our approach generates feasible two-stage stochastic AC-OPF solutions with comparable in-sample and out-of-sample feasibility, achieving a cost difference within 0.95% while providing up to two orders of magnitude speedup compared to iterative methods.

Beyond the two-stage stochastic AC-OPF problem, our DeepOPF-Stoc framework is also applicable to two-stage stochastic DC-OPF problems. Future research directions include: (i) Extending DeepOPF-Stoc to unsupervised learning schemes; (ii) Integrating adversarial training techniques or Bayesian neural networks into our framework to further improve robustness against unseen distributions; (iii) Extending our design to solve multi-stage stochastic AC-OPF problems; (iv) Extending our framework to stochastic AC-OPF problems with recourse mechanisms to recover feasibility when the second-stage subproblem is infeasible; and (v) Exploring the potential of CNN-/GNN-based approaches for solving the two-stage stochastic AC-OPF problem.

Acknowledgement

Commercial large language models were used for linguistic refinement in this paper. The authors would like to thank the anonymous reviewers for their helpful comments.

References

[1] J. Carpentier, "Contribution à l'étude du dispatching économique," *Bull. Soc. Francaise Elect.*, vol. 8, pp. 431–447, 1962.

[2] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, 2010.

[3] K. Baker, "Learning warm-start points for ac optimal power flow," in *Proc. IEEE MLSP*, 2019, pp. 1–6.

TABLE IV
SIMULATION RESULTS ON THE 793-BUS TEST SYSTEM.

Methods	Metrics	size=5	size=10	size=15
MIPS	C_{oos} (\$)	439,230	439,169	439,198
	η_{oos} (%)	93.6	95.6	96.5
	η_{IMS}	$\times 1$	$\times 1$	$\times 1$
	η_{AS}	$\times 1$	$\times 1$	$\times 1$
	t (s)	323	1148	2351
	VSS	9,869	9,783	8,082
Deterministic Model	C_{oos} (\$)	438,261	439,186	439,216
	η_{oos} (%)	84.7	85.8	88.4
	η_{IMS}	$\times 8$	$\times 29$	$\times 61$
	η_{AS}	$\times 9$	$\times 31$	$\times 63$
	t (s)	37	37	37
	VSS	0	0	0
Chance-constrained AC-OPF	C_{oos} (\$)	439,258	439,180	439,215
	η_{oos} (%)	98.6	98.8	98.8
	η_{IMS}	$\times 0.1$	$\times 0.2$	$\times 0.5$
	η_{AS}	$\times 0.1$	$\times 0.2$	$\times 0.5$
	t (s)	2580	4992	5122
	VSS	14,897	12,994	10,399
DeepOPF-Stoc	C_{oos} (\$)	440,082	441,199	441,461
	η_{oos} (%)	99.0	99.6	98.9
	η_{IMS} [1]	$\times 85$	$\times 156$	$\times 230$
	η_{AS} [1]	$\times 13$	$\times 13$	$\times 11$
	t (s)	25	87	212
	VSS	16,121	15,813	12,745

[1] The difference between η_{IMS} and η_{AS} reflects our method's performance variance across instances. Our method achieves a two orders of magnitude speedup for instances not requiring warm starts, but one order of magnitude when warm starts are necessary. η_{IMS} weights all instances equally, while η_{AS} inherently weights by runtime, resulting in the observed metric difference.

[4] Y. Zhou, B. Zhang, C. Xu, T. Lan, R. Diao, D. Shi, Z. Wang, and W.-J. Lee, "Deriving fast ac opf solutions via proximal policy optimization for secure and economic grid operation," *arXiv preprint arXiv:2003.12584*, 2020.

[5] P. L. Donti, D. Rolnick, and J. Z. Kolter, "DC3: A learning method for optimization with hard constraints," in *Proc. ICLR*, 2021.

[6] W. Huang and M. Chen, "DeepOPF-NGT: A Fast Unsupervised Learning Approach for Solving AC-OPF Problems without Ground Truth," in *Proc. ICML Workshop*, 2021.

[7] I. Mezghani, S. Misra, and D. Deka, "Stochastic ac optimal power flow: A data-driven approach," *Electr. Power Syst. Res.*, vol. 189, p. 106567, 2020.

[8] Y. Chen, Y. Wang, D. Kirschen, and B. Zhang, "Model-free renewable scenario generation using generative adversarial networks," *IEEE Trans. Power Syst.*, vol. 33, no. 3, pp. 3265–3275, 2018.

[9] Z. Zhou, C. Liu, and A. Botterud, "Stochastic methods applied to power system operations with renewable energy: A review," *Tech. Rep. ANL/ESD -16/14 Argonne Nat. Lab, Argonne, IL, USA*, 2016.

[10] Y. Ng, S. Misra, L. A. Roald, and S. Backhaus, "Statistical learning for DC optimal power flow," in *Proc. PSCC*, 2018, pp. 1–7.

[11] F. Diehl, "Warm-starting ac optimal power flow with graph neural networks," in *Proc. NeurIPS*, 2019, pp. 1–6.

[12] W. Dong, Z. Xie, G. Kestor, and D. Li, "Smart-pgsim: Using neural network to accelerate ac-opf power grid simulation," *arXiv preprint arXiv:2008.11827*, 2020.

[13] A. Robson, M. Jamei, C. Ududec, and L. Mones, "Learning an optimally reduced formulation of opf through meta-optimization," *arXiv preprint arXiv:1911.06784*, 2019.

[14] K. Baker, "A Learning-boosted Quasi-Newton Method for AC Optimal Power Flow," *arXiv preprint arXiv:2007.06074*, 2020.

[15] D. Biagioni, P. Graf, X. Zhang, A. S. Zamzam, K. Baker, and J. King, "Learning-accelerated admm for distributed dc optimal power flow," *IEEE Control Syst. Lett.*, vol. 6, pp. 1–6, 2020.

[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 Sys. J.*, vol. 17, no. 1, pp. 673–683, 2022.

[17] A. Velloso and P. Van Hentenryck, "Combining deep learning and optimization for preventive security-constrained dc optimal power flow," *IEEE Trans. Power Syst.*, vol. 36, no. 4, pp. 3618–3628, 2021.

[18] A. S. Zamzam and K. Baker, "Learning optimal solutions for extremely fast ac optimal power flow," in *Proc. IEEE SmartGridComm*, 2020, pp. 1–6.

[19] M. Gao, J. Yu, Z. Yang, and J. Zhao, "A physics-guided graph convolution neural network for optimal power flow," *IEEE Trans. Power Syst.*, vol. 39, no. 1, pp. 380–390, 2023.

[20] Y. Jia, X. Bai, L. Zheng, Z. Weng, and Y. Li, "Convopf-dop: A data-driven method for solving ac-opf based on cnn considering different operation patterns," *IEEE Trans. Power Syst.*, vol. 38, no. 1, pp. 853–860, 2022.

[21] T. Zhao, X. Pan, M. Chen, A. Venzke, and S. H. Low, "Deepopf+: A deep neural network approach for dc optimal power flow for ensuring feasibility," in *Proc. IEEE SmartGridComm*, 2020, pp. 1–6.

[22] M. Li, S. Kolouri, and J. Mohammadi, "Learning to solve optimization problems with hard linear constraints," *IEEE Access*, vol. 11, pp. 59 995–60 004, 2023.

[23] L. Zhang, D. Tabas, and B. Zhang, "An efficient learning-based solver for two-stage dc optimal power flow with feasibility guarantees," *arXiv preprint arXiv:2304.01409*, 2023.

[24] C. E. Murillo-Sanchez, R. D. Zimmerman, C. L. Anderson, and R. J. Thomas, "A stochastic, contingency-based security-constrained optimal power flow for the procurement of energy and distributed reserve," *Decis. Support Syst.*, vol. 56, pp. 1–10, 2013.

[25] A. Lorca and X. A. Sun, "The adaptive robust multi-period alternating current optimal power flow problem," *IEEE Trans. Power Syst.*, vol. 33, no. 2, pp. 1993–2003, 2017.

[26] R. A. Jabr, "Adjustable robust opf with renewable energy sources," *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 4742–4751, 2013.

[27] R. Louca and E. Bitar, "Robust ac optimal power flow," *IEEE Trans. Power Syst.*, vol. 34, no. 3, pp. 1669–1681, 2018.

[28] D. Lee, K. Turitsyn, D. K. Molzahn, and L. A. Roald, "Robust ac optimal power flow with robust convex restriction," *IEEE Trans. Power Syst.*, vol. 36, no. 6, pp. 4953–4966, 2021.

[29] L. Roald and G. Andersson, "Chance-constrained ac optimal power flow: Reformulations and efficient algorithms," *IEEE Trans. Power Syst.*, vol. 33, no. 3, pp. 2906–2918, 2017.

[30] D. Bienstock, M. Chertkov, and S. Harnett, "Chance-constrained optimal power flow: Risk-aware network control under uncertainty," *Siam Review*, vol. 56, no. 3, pp. 461–495, 2014.

[31] T. Mühlpfordt, L. Roald, V. Hagenmeyer, T. Faulwasser, and S. Misra, "Chance-constrained ac optimal power flow: A polynomial chaos approach," *IEEE Trans. Power Syst.*, vol. 34, no. 6, pp. 4806–4816, 2019.

[32] A. Agarwal, P. L. Donti, J. Z. Kolter, and L. Pileggi, "Employing adversarial robustness techniques for large-scale stochastic optimal power flow," *Electr. Power Syst. Res.*, vol. 212, p. 108497, 2022.

[33] D. Phan and S. Ghosh, "Two-stage stochastic optimization for optimal power flow under renewable generation uncertainty," *Proc. TOMACS*, vol. 24, no. 1, pp. 1–22, 2014.

[34] F. Hasan, A. Kargarian, and J. Mohammadi, "Hybrid learning aided inactive constraints filtering algorithm to enhance ac opf solution time," *IEEE Trans. Ind. Appl.*, vol. 57, no. 2, pp. 1325–1334, 2021.

[35] S. Liu, Y. Guo, W. Tang, H. Sun, and W. Huang, "Predicting active constraints set in security-constrained optimal power flow via deep neural network," in *Proc. PESGM*. IEEE, 2021, pp. 01–05.

[36] S. Misra, L. Roald, and Y. Ng, "Learning for constrained optimization: Identifying optimal active constraint sets," *INFORMS J. Comput.*, vol. 34, no. 1, pp. 463–480, 2022.

[37] L. Chen and J. E. Tate, "Hot-starting the ac power flow with convolutional neural networks," *arXiv preprint arXiv:2004.09342*, 2020.

[38] T. Chen, X. Chen, W. Chen, Z. Wang, H. Heaton, J. Liu, and W. Yin, "Learning to optimize: A primer and a benchmark," *J. Mach. Learn. Res.*, vol. 23, no. 1, pp. 8562–8620, 2022.

[39] W. Huang, X. Pan, M. Chen, and S. H. Low, "Deepopf-v: Solving ac-opf problems efficiently," *IEEE Trans. Power Syst.*, vol. 37, no. 1, pp. 800–803, 2021.

[40] M. Chatzos, F. Fioretto, T. W. Mak, and P. Van Hentenryck, "High-fidelity machine learning approximations of large-scale optimal power flow," *arXiv preprint arXiv:2006.16356*, 2020.

[41] E. Liang, M. Chen, and S. H. Low, "Low complexity homeomorphic projection to ensure neural-network solution feasibility for optimization over (non-)convex set," in *Proc. ICML*, 2023.

[42] F. Fioretto, T. W. Mak, and P. Van Hentenryck, "Predicting AC optimal power flows: Combining deep learning and lagrangian dual methods," in *Proc. AAAI*, vol. 34, no. 01, 2020, pp. 630–637.

[43] F. Fioretto, P. V. Hentenryck, T. W. Mak, C. Tran, F. Baldo, and M. Lombardi, "Lagrangian duality for constrained deep learning," in *Proc. ECML PKDD*. Springer, 2020, pp. 118–135.

[44] Y. Chen, L. Zhang, and B. Zhang, "Learning to solve dcopf: A duality approach," *Electr. Power Syst. Res.*, vol. 213, p. 108595, 2022.

[45] M. K. Singh, V. Kekatos, and G. B. Giannakis, "Learning to solve the ac-opf using sensitivity-informed deep neural networks," *IEEE Trans. Power Syst.*, vol. 37, no. 4, pp. 2833–2846, 2021.

[46] R. Nellikkath and S. Chatzivasileiadis, "Physics-informed neural networks for ac optimal power flow," *Electr. Power Syst. Res.*, vol. 212, p. 108412, 2022.

[47] X. Lei, Z. Yang, J. Yu, J. Zhao, Q. Gao, and H. Yu, "Data-driven optimal power flow: A physics-informed machine learning approach," *IEEE Trans. Power Syst.*, vol. 36, no. 1, pp. 346–354, 2020.

[48] M. Kim and H. Kim, "Self-supervised equality embedded deep lagrange dual for approximate constrained optimization," *arXiv preprint arXiv:2306.06674*, 2023.

[49] X. Pan, W. Huang, M. Chen, and S. H. Low, "Deepopf-al: Augmented learning for solving ac-opf problems with multiple load-solution mappings," *arXiv preprint arXiv:2206.03365*, 2022.

[50] J. Wang and P. Srikantha, "Fast optimal power flow with guarantees via an unsupervised generative model," *IEEE Trans. Power Syst.*, vol. 38, no. 5, pp. 4593–4604, 2022.

[51] T. Wu, A. Scaglione, and D. Arnold, "Constrained reinforcement learning for stochastic dynamic optimal power flow control," *arXiv preprint arXiv:2302.10382*, 2023.

[52] K. Chen, S. Bose, and Y. Zhang, "Unsupervised deep learning for ac optimal power flow via lagrangian duality," in *Proc. GLOBECOM*, 2022, pp. 5305–5310.

[53] N. Guha, Z. Wang, M. Wytock, and A. Majumdar, "Machine learning for ac optimal power flow," *arXiv preprint arXiv:1910.08842*, 2019.

[54] T. Zhao, X. Pan, M. Chen, and S. Low, "Ensuring dnn solution feasibility for optimization problems with linear constraints," in *Proc. ICLR*, 2023.

[55] E. Liang and M. Chen, "Efficient bisection projection to ensure neural-network solution feasibility for optimization over general set," in *Proc. ICML*. PMLR, 2025.

[56] S. Gupta, S. Misra, D. Deka, and V. Kekatos, "Dnn-based policies for stochastic ac opf," *Electr. Power Syst. Res.*, vol. 213, p. 108563, 2022.

[57] M. Mitrovic, A. Lukashevich, P. Vorobev, V. Terzija, S. Budennyy, Y. Maximov, and D. Deka, "Data-driven stochastic ac-opf using gaussian process regression," *Int. J. Electr. Power. Energy. Syst.*, vol. 152, p. 109249, 2023.

[58] Y. Bengio, E. Frejinger, A. Lodi, R. Patel, and S. Sankaranarayanan, "A learning-based algorithm to quickly compute good primal solutions for stochastic integer programs," in *Proc. CPAIOR*, 2020.

[59] J. Keutchayan, J. Ortmann, and W. Rei, "Problem-driven scenario clustering in stochastic optimization," *Comput. Manag. Sci.*, vol. 20, no. 1, p. 13, 2023.

[60] V. Nair, D. Dvijotham, I. Dunning, and O. Vinyals, "Learning fast optimizers for contextual stochastic integer programs." in *UAI*, 2018, pp. 591–600.

[61] S. Ghadimi, R. T. Perkins, and W. B. Powell, "Reinforcement learning via parametric cost function approximation for multistage stochastic programming," *arXiv preprint arXiv:2001.00831*, 2020.

[62] J. Dumouchelle, R. Patel, E. B. Khalil, and M. Bodur, "Neur2sp: Neural two-stage stochastic programming," 2022.

[63] G. C. Calafiore and M. C. Campi, "The scenario approach to robust control design," *IEEE Trans. Autom. Control,*, vol. 51, no. 5, pp. 742–753, 2006.

[64] T. Yong and R. Lasseter, "Stochastic optimal power flow: formulation and solution," in *Proc. Power Engineering Society Summer Meeting*, vol. 1. IEEE, 2000, pp. 237–242.

[65] X. Pan, T. Zhao, and M. Chen, "Deepopf: Deep neural network for DC optimal power flow," in *IEEE SmartGridComm*, Beijing, China, Oct. 2019.

[66] J. Schmidt-Hieber, "The kolmogorov–arnold representation theorem revisited," *Neural Netw.*, vol. 137, pp. 119–126, 2021.

[67] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," *Proc. NeurIPS*, p. 3394–3404, 2017.

[68] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[69] S. Babaeinejadsarookolaee, A. Birchfield, R. D. Christie, C. Coffrin, C. DeMarco, R. Diao, M. Ferris, S. Fliscounakis, S. Greene, R. Huang *et al.*, "The power grid library for benchmarking ac optimal power flow algorithms," *arXiv preprint arXiv:1908.02788*, 2019.

[70] M. Zhou, "Test cases for DeepOPF-Stoc," GitHub, 2024. [Online]. Available:https://github.com/Mzhou-cityu/DeepOPF-Stoc.

[71] J. Birge and F. Louveaux, *Introduction to Stochastic Programming*. New York, NY, USA: Springer Verlag, 1997.

[72] Z. Yan and Y. Xu, "Real-time optimal power flow: A lagrangian based deep reinforcement learning approach," *IEEE Trans. Power Syst.*, vol. 35, no. 4, pp. 3270–3273, 2020.

[73] P. Wang, S. Yang, S. Li, Z. Wang, and P. Li, "Polynomial width is sufficient for set representation with high-dimensional features," *arXiv preprint arXiv:2307.04001*, 2023.

APPENDIX A
MATHEMATICAL DEFINITION OF THE EVALUATION METRICS

Let n denote the number of test instances, and K the number of out-of-sample scenarios per instance. For each test instance $j \in \{1, \ldots, n\}$ and scenario $k \in \{1, \ldots, K\}$, we first define $C_{1,j}$ as the first-stage cost for instance j and $C_{2,j,k}$ as the second-stage cost for instance j under scenario k. If the second-stage subproblem is infeasible for scenario k of instance j, $C_{2,j,k}$ is set to 0. We also define $\mathbb{I}_{j,k}$ as an indicator that equals 1 if the second-stage subproblem is feasible for instance j under scenario k, and 0 otherwise. Based on the above definitions, we define the evaluation metrics as follows:

Out-of-sample cost (C_{oos}): the out-of-sample cost is the sum of the first-stage generation cost and the average second-stage cost over feasible out-of-sample scenarios:

$$C_{\text{oos}} = \frac{1}{n} \sum_{j=1}^{n} \left[C_{1,j} + \frac{1}{\sum_{k=1}^{K} \mathbb{I}_{j,k}} \sum_{k=1}^{K} \mathbb{I}_{j,k} \cdot C_{2,j,k} \right].$$

If a scenario leads to infeasibility in the second-stage subproblem, it is excluded from the averaging process in the calculation of the out-of-sample cost. Based on the out-of-sample cost, we can calculate the cost difference with respect to the MIPS solution as follows:

$$C_{\text{dif}} = \frac{C_{j,\text{oos}}^{\text{method}} - C_{j,\text{oos}}^{\text{MIPS}}}{C_{j,\text{oos}}^{\text{MIPS}}}.$$

where $C_{j,\text{oos}}^{\text{method}}$ and $C_{j,\text{oos}}^{\text{MIPS}}$ denote the out-of-sample cost for instance j using the evaluated method and MIPS, respectively.

Out-of-sample feasibility rate (η_{oos}): The out-of-sample feasibility rate measures the proportion of out-of-sample scenarios for which a feasible second-stage solution exists, which is calculated as follows:

$$\eta_{\text{oos}} = \frac{1}{nK} \sum_{j=1}^{n} \sum_{k=1}^{K} \mathbb{I}_{j,k}.$$

Runtime (t): The runtime is the execution time of the evaluated method for solving the two-stage stochastic AC-OPF problem. Based on this, we calculate two complementary speedup metrics to provide a comprehensive performance evaluation.

The instance-wise mean speedup (η_{IMS}) captures the average of individual instance speedups, which provides insight into the method's relative performance across diverse test instances:

$$\eta_{\text{IMS}} = \frac{1}{n} \sum_{j=1}^{n} \frac{t_j^{\text{MIPS}}}{t_j^{\text{method}}}.$$

where t_j^{method} and t_j^{MIPS} denote the runtime for instance j using the evaluated method and MIPS, respectively.

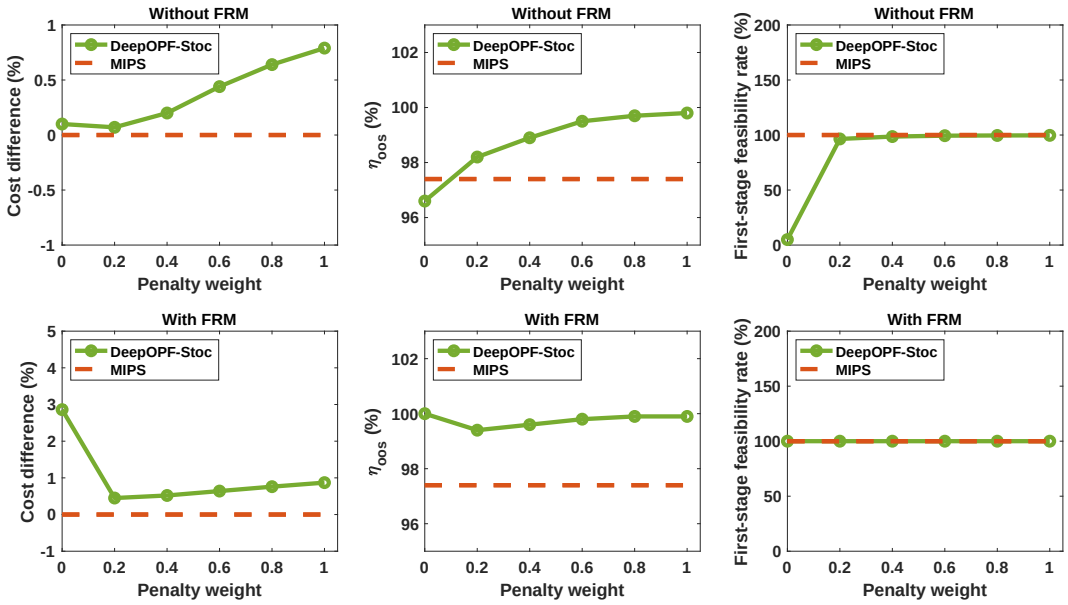


Fig. 5. Performance of DeepOPF-Stoc under different penalty weights.

The aggregate speedup (η_{AS}) represents the ratio of total runtimes across all instances, which reflects the overall computational savings when processing large batches of instances:

$$\eta_{\text{AS}} = \frac{\sum_{j=1}^{n} t_j^{\text{MIPS}}}{\sum_{j=1}^{n} t_j^{\text{method}}}.$$

Value of the stochastic solution (VSS): We propose the following generalized VSS metric that accounts for both cost and feasibility considerations:

$$\text{VSS} = \left(C_{\text{oos}}^{\text{det}} - \lambda \cdot \eta_{\text{oos}}^{\text{det}}\right) - \left(C_{\text{oos}}^{\text{method}} - \lambda \cdot \eta_{\text{oos}}^{\text{method}}\right),$$

where $C_{\text{oos}}^{\text{det}}$ and $C_{\text{oos}}^{\text{method}}$ represent out-of-sample costs for the deterministic model and our method respectively, while $\eta_{\text{oos}}^{\text{det}}$ and $\eta_{\text{oos}}^{\text{method}}$ are their corresponding out-of-sample feasibility rates. The parameter $\lambda \geq 0$ weights feasibility relative to cost (we use $\lambda = 10^5$ in all simulations). This formulation provides a comprehensive assessment of stochastic solution benefits by simultaneously evaluating cost efficiency and constraint satisfaction performance.

Appendix B
Analysis of Out-of-Sample Feasibility Performance

In our simulation, we observed that DeepOPF-Stoc achieves a higher out-of-sample feasibility rate than MIPS. We understand that this phenomenon is due to the different optimization behaviors of the two approaches. MIPS, as a cost-minimizing solver, tends to produce solutions positioned near the boundary of the feasible region to achieve optimal economic performance. While this strategy is economically efficient, it can result in reduced robustness when facing out-of-sample uncertainty realizations. In contrast, our DeepOPF-Stoc approach incorporates constraint enforcement mechanisms that guide solutions toward more conservative positions within the feasible region, trading modest increases in generation costs for enhanced operational robustness.

To provide rigorous empirical validation of this explanation, we conducted additional ablation experiments by removing the constraint enforcement mechanism on the IEEE 118-bus test system with 25 in-sample scenarios, where uncertainty in load and renewable generation is modeled as the Gaussian distribution with a medium uncertainty level scheme described in Section VI-A. The results in Table V demonstrate that DeepOPF-Stoc without constraint enforcement, indeed, achieves lower out-of-sample feasibility rates than MIPS, thereby confirming our hypothesis. This comparison clearly establishes that the constraint enforcement mechanism is the key component responsible for DeepOPF-Stoc's superior out-of-sample feasibility performance.

TABLE V
Out-of-Sample Feasibility Rates under Different Settings.

Setting	With constraint enforcement	Without constraint enforcement	MIPS
η_{oos} (%)	99.8	96.6	97.4

Appendix C
Performance with Different Penalty Weights

We conduct simulations on the IEEE 118-bus test system with 25 in-sample scenarios with uncertainty modeled as the Gaussian distribution with a medium uncertainty level described in Section VI-A, to systematically investigate whether and how the trade-off between cost efficiency and solution feasibility can be dynamically controlled through penalty weight adjustments in the loss function ((18), page 6). We designed comprehensive experiments with two complementary configurations.

Isolated penalty weight analysis (without FRM): To isolate the direct effect of penalty weights, we first evaluated DeepOPF-Stoc performance across different penalty weight values with the feasibility recovery mechanism (FRM) disabled. The results in Figure 5 demonstrate a clear and controllable trade-off relationship: increasing penalty weights improves out-of-sample feasibility rates at the cost of slightly higher generation costs, while decreasing penalty weights enhances cost performance but reduces feasibility. This confirms that penalty weights function as an effective tuning mechanism for directly controlling the feasibility-optimality balance.

Complete framework analysis (with FRM enabled): We conducted a second experimental series to understand how penalty weights indirectly influence cost efficiency when feasibility is guaranteed through our full DeepOPF-Stoc framework. As expected, as shown in Figure 5, feasibility rates remain consistently high across all penalty weight settings due to the corrective function of FRM. Interestingly, the cost difference exhibits a U-shaped pattern as penalty weights vary, which can be explained through three distinct operational regimes:

(i) *Low penalty weights*: PPNN frequently generates infeasible outputs, necessitating substantial feasibility restoration operations that result in significant cost increases.

(ii) *Moderate penalty weights*: PPNN learns to produce solutions that balance proximity to the feasible region with cost minimization objectives, achieving both high feasibility and low cost differences.

(iii) *High penalty weights*: The penalty term dominates the loss function, yielding conservative solutions with excellent feasibility but compromised economic efficiency.

These complementary experimental analyses establish that penalty weights provide effective control over the feasibility-optimality trade-off, offering system operators the flexibility to adapt solution characteristics to varying operational priorities.

APPENDIX D
PROOF OF PROPOSITION 1

Proof. We aim to prove that the input to first-stage solution mapping $\mathcal{P}^*$ in the two-stage stochastic AC-OPF problem is partially permutation-invariant with respect to the second-stage scenarios. For notational convenience, we express the scenario-based two-stage stochastic AC-OPF problem in the following compact form:

$$\min \; F(\boldsymbol{\zeta}_0) + \frac{1}{K}\sum_{k=1}^{K} F(\boldsymbol{\zeta}_k) \tag{20}$$

$$\text{s.t. } G(\boldsymbol{\zeta}_k, \boldsymbol{d}_k) = 0, \;\; k = 0, \ldots, K, \tag{21}$$

$$H(\boldsymbol{\zeta}_k, \boldsymbol{d}_k) \leq 0, \;\; k = 0, \ldots, K, \tag{22}$$

$$|\boldsymbol{\zeta}_k - \boldsymbol{\zeta}_0| \leq \Delta, \;\; k = 1, \ldots, K, \tag{23}$$

$$\text{var. } \boldsymbol{\zeta}_k, \;\; k = 0, \ldots, K. \tag{24}$$

where $\boldsymbol{\zeta}_k = (p^g_{ik}, q^g_{ik}, v_{ik}, \theta_{ik})_{i\in\mathcal{B}}$ and $\boldsymbol{d}_k$ are the decision variables and load inputs at scenario k, respectively. $F(\cdot)$ corresponds to the quadratic generation cost function in (1), and constraints (21)-(23) denote power balance equations in (2)-(5), operational limits in (6)-(10), and ramping constraints in (11), respectively.

Assumption: For any input $(\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_K)$ from the load domain, the two-stage stochastic AC-OPF problem has a *unique* optimal solution.

Let $(\boldsymbol{\zeta}^*_0, \boldsymbol{\zeta}^*_1, \ldots, \boldsymbol{\zeta}^*_K)$ be the unique optimal solution for load input $(\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_K)$. For any permutation π of scenario indices $\{1, \ldots, K\}$, we need to show that $\boldsymbol{\zeta}^*_0$ remains unchanged when the second-stage scenarios are permuted.

Step 1 (Feasibility): For the permuted input $(\boldsymbol{d}_0, \boldsymbol{d}_{\pi_1}, \ldots, \boldsymbol{d}_{\pi_K})$, the solution $(\boldsymbol{\zeta}^*_0, \boldsymbol{\zeta}^*_{\pi_1}, \ldots, \boldsymbol{\zeta}^*_{\pi_K})$ is feasible because:

- The first-stage constraints are identical: $G(\boldsymbol{\zeta}^*_0, \boldsymbol{d}_0) = 0$ and $H(\boldsymbol{\zeta}^*_0, \boldsymbol{d}_0) \leq 0$.
- Since π is a complete rearrangement of the integers 1 to K in a specific order, we have $\pi_k \in \{1, \ldots, K\}$ for all $k \in \{1, \ldots, K\}$. Thus, the second-stage constraints are satisfied: $G(\boldsymbol{\zeta}^*_{\pi_k}, \boldsymbol{d}_{\pi_k}) = 0$ and $H(\boldsymbol{\zeta}^*_{\pi_k}, \boldsymbol{d}_{\pi_k}) \leq 0$ for $k = 1, \ldots, K$. For the same reason, the coupling constraints remain valid: $|\boldsymbol{\zeta}^*_{\pi_k} - \boldsymbol{\zeta}^*_0| \leq \Delta$ for $k = 1, \ldots, K$.

Step 2 (Optimality by Contradiction): Suppose there exists a different solution $(\boldsymbol{\zeta}'_0, \boldsymbol{\zeta}'_1, \ldots, \boldsymbol{\zeta}'_K)$ that is feasible for the permuted input $(\boldsymbol{d}_0, \boldsymbol{d}_{\pi_1}, \ldots, \boldsymbol{d}_{\pi_K})$ and has a lower objective value than $(\boldsymbol{\zeta}^*_0, \boldsymbol{\zeta}^*_{\pi_1}, \ldots, \boldsymbol{\zeta}^*_{\pi_K})$:

$$F(\boldsymbol{\zeta}'_0) + \frac{1}{K}\sum_{k=1}^{K} F(\boldsymbol{\zeta}'_k) < F(\boldsymbol{\zeta}^*_0) + \frac{1}{K}\sum_{k=1}^{K} F(\boldsymbol{\zeta}^*_{\pi_k}). \tag{25}$$

Let π^{-1} denote the inverse permutation of π, such that if $j = \pi_k$, then ${\pi^{-1}}_j = k$ for all $k, j \in \{1, \ldots, K\}$. By the same reasoning as Step 1, solution $(\boldsymbol{\zeta}'_0, \boldsymbol{\zeta}'_{{\pi^{-1}}_1}, \ldots, \boldsymbol{\zeta}'_{{\pi^{-1}}_K})$ is feasible for the original input $(\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_K)$. Furthermore:

$$\begin{aligned} F(\boldsymbol{\zeta}'_0) + \frac{1}{K}\sum_{k=1}^{K} F(\boldsymbol{\zeta}'_{{\pi^{-1}}_k}) &= F(\boldsymbol{\zeta}'_0) + \frac{1}{K}\sum_{k=1}^{K} F(\boldsymbol{\zeta}'_k) \\ &< F(\boldsymbol{\zeta}^*_0) + \frac{1}{K}\sum_{k=1}^{K} F(\boldsymbol{\zeta}^*_{\pi_k}) \\ &= F(\boldsymbol{\zeta}^*_0) + \frac{1}{K}\sum_{k=1}^{K} F(\boldsymbol{\zeta}^*_k). \end{aligned} \tag{26}$$

Thus, we find a feasible solution $(\boldsymbol{\zeta}'_0, \boldsymbol{\zeta}'_{{\pi^{-1}}_1}, \ldots, \boldsymbol{\zeta}'_{{\pi^{-1}}_K})$ for input $(\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_K)$ with a lower generation cost than $(\boldsymbol{\zeta}^*_0, \boldsymbol{\zeta}^*_1, \ldots, \boldsymbol{\zeta}^*_K)$. This contradicts the assumption that $(\boldsymbol{\zeta}^*_0, \boldsymbol{\zeta}^*_1, \ldots, \boldsymbol{\zeta}^*_K)$ is the *unique* optimal solution for the original input $(\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_K)$. Therefore, $(\boldsymbol{\zeta}^*_0, \boldsymbol{\zeta}^*_{\pi_1}, \ldots, \boldsymbol{\zeta}^*_{\pi_K})$ must be optimal for the permuted input. By the uniqueness assumption, it is the unique optimal solution. Hence, the first-stage solution $\boldsymbol{\zeta}^*_0$ remains invariant to permutations of second-stage scenarios.

Recall that the mapping $\mathcal{P}^*$ maps $(\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_K)$ to the optimal first-stage decision $\boldsymbol{\zeta}^*_0$ for the two-stage stochastic AC-OPF problem. Since we have shown that $\boldsymbol{\zeta}^*_0$ remains the same when second-stage scenarios are permuted, we have:

$$\mathcal{P}^*(\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_K) = \mathcal{P}^*(\boldsymbol{d}_0, \pi(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K)), \tag{27}$$

for any permutation π of $\{1, \ldots, K\}$. This proves the partial permutation-invariance property of $\mathcal{P}^*$.

□

APPENDIX E
PROOF OF THEOREM 1

Proof. We aim to prove that for the partially permutation-invariant mapping $\mathcal{P}^* : \mathbb{R}^{m(K+1)} \to \mathbb{R}^n$, there exist two continuous mappings $\phi : \mathbb{R}^m \to \mathbb{R}^{b_\phi}$ and $\rho : \mathbb{R}^{m+b_\phi} \to \mathbb{R}^n$ such that $\mathcal{P}^*(\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_K) = \rho\left(\boldsymbol{d}_0, \sum_{k=1}^K \phi(\boldsymbol{d}_k)\right)$, where $\boldsymbol{d}_k \in \mathbb{R}^m$ for all $k \in \{0, \ldots, K\}$ and the embedding dimension is bounded by $b_\phi \leq K^5 m^2$.

First, we define a power mapping of degree K that transforms a scalar into a vector of its powers:

$$\psi_K : \mathbb{R} \to \mathbb{R}^K, \quad \psi_K(z) = [z \quad z^2 \quad \ldots \quad z^K]^T. \tag{28}$$

Using the power mapping, we construct a feature transformation function $\phi : \mathbb{R}^m \to \mathbb{R}^{NK}$ that applies multiple linear projections followed by power mappings:

$$\phi(\boldsymbol{x}) = [\psi_K(\boldsymbol{w}_1^T \boldsymbol{x}), \psi_K(\boldsymbol{w}_2^T \boldsymbol{x}), \ldots, \psi_K(\boldsymbol{w}_N^T \boldsymbol{x})], \tag{29}$$

where $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N \in \mathbb{R}^m$ are linear weights.

We say two inputs $(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K)$, $(\boldsymbol{d}'_1, \ldots, \boldsymbol{d}'_K)$ are equivalent if there exists a permutation π of $\{1, \ldots, K\}$, such that $(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K) = \pi(\boldsymbol{d}'_1, \ldots, \boldsymbol{d}'_K)$. We denote this equivalence as $(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K) \sim (\boldsymbol{d}'_1, \ldots, \boldsymbol{d}'_K)$.

According to Theorem 3.1 of [73], we can construct a set of N linear weights $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N$ with $N \leq K^4 m^2$ and a summation embedding $\Phi : \mathbb{R}^{mK} \to \mathbb{R}^{NK}$:

$$\Phi(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K) = \sum_{k=1}^K \phi(\boldsymbol{d}_k), \tag{30}$$

such that, for all $(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K)$, $(\boldsymbol{d}'_1, \ldots, \boldsymbol{d}'_K)$, $\Phi(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K) = \Phi(\boldsymbol{d}'_1, \ldots, \boldsymbol{d}'_K)$ implies $(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K) \sim (\boldsymbol{d}'_1, \ldots, \boldsymbol{d}'_K)$. Furthermore, there exists a continuous function $\Phi^{-1} : \mathbb{R}^{NK} \to \mathbb{R}^{mK}$, such that $\Phi^{-1} \circ \Phi(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K) \sim (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K)$ [73].

Leveraging this result, we can construct a universal representation function ρ as:

$$\rho(\boldsymbol{d}_0, \boldsymbol{s}) = \mathcal{P}^*(\boldsymbol{d}_0, \Phi^{-1}(\boldsymbol{s})), \tag{31}$$

where $\mathcal{P}^*$ is our target mapping, $\boldsymbol{d}_0 \in \mathbb{R}^m$ and $\boldsymbol{s} \in \mathbb{R}^{NK}$. Based on this construction, we have:

$$\begin{aligned}
\rho\left(\boldsymbol{d}_0, \sum_{k=1}^K \phi(\boldsymbol{d}_k)\right) &= \mathcal{P}^*\left(\boldsymbol{d}_0, \Phi^{-1}\left(\sum_{k=1}^K \phi(\boldsymbol{d}_k)\right)\right) \\
&= \mathcal{P}^*\left(\boldsymbol{d}_0, \Phi^{-1} \circ \Phi(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K)\right) \\
&= \mathcal{P}^*(\boldsymbol{d}_0, \boldsymbol{d}'_1, \ldots, \boldsymbol{d}'_K),
\end{aligned}$$

where $(\boldsymbol{d}'_1, \ldots, \boldsymbol{d}'_K) \sim (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K)$. Based on the partial permutation-invariance property of $\mathcal{P}^*$, we have:

$$\begin{aligned}
\rho\left(\boldsymbol{d}_0, \sum_{k=1}^K \phi(\boldsymbol{d}_k)\right) &= \mathcal{P}^*(\boldsymbol{d}_0, \boldsymbol{d}'_1, \ldots, \boldsymbol{d}'_K) \\
&= \mathcal{P}^*(\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_K).
\end{aligned} \tag{32}$$

The resulting embedding dimension is $b_\phi = K \cdot N$. As $N \leq K^4 m^2$, we have $b_\phi \leq K^5 m^2$.

□

APPENDIX F
PROOF OF THEOREM 2

Proof. We aim to prove that, for any continuous mappings ρ^* and ϕ^* that achieve the infimum of $\epsilon(b_\phi)$, there exists a PPNN composed of neural networks ρ_{nn} and ϕ_{nn}, such that its approximation error to $\mathcal{P}^*$ is bounded by:

$$\epsilon(b_\phi) \leq \epsilon_p \leq \epsilon(b_\phi) + K \cdot \text{Lip}(\rho^*)\epsilon_{\phi_{\text{nn}}} + \epsilon_{\rho_{\text{nn}}}. \tag{33}$$

where

$$\epsilon(b_\phi) \triangleq \inf_{\substack{\phi \in \mathcal{F} \\ \rho \in \mathcal{Q}}} \sup_{\boldsymbol{l} \in \mathcal{D}} \left\| \mathcal{P}^*(\boldsymbol{l}) - \rho\left(\boldsymbol{d}_0, \sum_{k=1}^K \phi(\boldsymbol{d}_k)\right) \right\|_2, \tag{34}$$

$$\epsilon_{\text{p}} \triangleq \inf_{\substack{\phi \in \mathcal{H} \\ \rho \in \mathcal{V}}} \sup_{\boldsymbol{l} \in \mathcal{D}} \left\| \mathcal{P}^*(\boldsymbol{l}) - \rho\left(\boldsymbol{d}_0, \sum_{k=1}^K \phi(\boldsymbol{d}_k)\right) \right\|_2, \tag{35}$$

$$\epsilon_{\phi_{\text{nn}}} \triangleq \inf_{\phi \in \mathcal{H}} \sup_{\boldsymbol{x} \in \mathcal{X}} \left\| \phi^*(\boldsymbol{x}) - \phi(\boldsymbol{x}) \right\|_2, \tag{36}$$

$$\epsilon_{\rho_{\text{nn}}} \triangleq \inf_{\rho \in \mathcal{V}} \sup_{\boldsymbol{y} \in \mathcal{Y}} \left\| \rho^*(\boldsymbol{y}) - \rho(\boldsymbol{y}) \right\|_2. \tag{37}$$

$\mathcal{F}$ denotes the space of all continuous mappings $\mathbb{R}^m \to \mathbb{R}^{b_\phi}$. $\mathcal{Q}$ represents the space of all continuous mappings $\mathbb{R}^{m+b_\phi} \to \mathbb{R}^n$. $\mathcal{H}$ comprises mappings implementable by neural networks ϕ_{nn} with architecture $\mathbb{R}^m \to \mathbb{R}^{b_\phi}$. $\mathcal{V}$ consists of mappings implementable by neural networks ρ_{nn} with architecture $\mathbb{R}^{m+b_\phi} \to \mathbb{R}^n$. ϕ^* and ρ^* are a pair of optimal continuous mappings that achieve the infimum in $\epsilon(b_\phi)$ with domains $\mathcal{X} = \mathbb{R}^m$ and $\mathcal{Y} = \mathbb{R}^{m+b_\phi}$, respectively. ϕ^*_{nn} is a mapping that achieves the infimum of $\epsilon_{\phi_{\text{nn}}}$. $\boldsymbol{l} = (\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_K) \in \mathcal{D}$ is the input to the two-stage stochastic AC-OPF problem, with $\boldsymbol{d}_k \in \mathbb{R}^m$ for $k = 0, \ldots, K$.

Part 1. Lower bound: We prove that $\epsilon(b_\phi) \leq \epsilon_p$ by analyzing the relationship between the function spaces of continuous mappings and their neural network approximations.

Since neural networks implement a subset of continuous functions, we have the set inclusions $\mathcal{H} \subseteq \mathcal{F}$ and $\mathcal{V} \subseteq \mathcal{Q}$. Consequently, ϵ_p computes its infimum over a more restricted domain than $\epsilon(b_\phi)$, yielding the inequality:

$$\epsilon(b_\phi) \leq \epsilon_p. \tag{38}$$

Part 2. Upper bound: To establish the upper bound, we introduce the optimal continuous mappings ρ^* and ϕ^* that achieve the infimum in $\epsilon(b_\phi)$.

Step 2.1: For any $\phi \in \mathcal{H}$ and $\rho \in \mathcal{V}$, applying the triangle inequality:

$$\begin{aligned}
&\left\| \mathcal{P}^*(\boldsymbol{l}) - \rho\left(\boldsymbol{d}_0, \sum_{k=1}^K \phi(\boldsymbol{d}_k)\right) \right\|_2 \\
&\leq \left\| \mathcal{P}^*(\boldsymbol{l}) - \rho^*\left(\boldsymbol{d}_0, \sum_{k=1}^K \phi^*(\boldsymbol{d}_k)\right) \right\|_2 \\
&\quad + \left\| \rho^*\left(\boldsymbol{d}_0, \sum_{k=1}^K \phi^*(\boldsymbol{d}_k)\right) - \rho\left(\boldsymbol{d}_0, \sum_{k=1}^K \phi(\boldsymbol{d}_k)\right) \right\|_2.
\end{aligned}$$

Step 2.2: Taking the supremum over all $\boldsymbol{l} \in \mathcal{D}$ and applying the property that:

$$\sup_{\boldsymbol{l} \in \mathcal{D}} (U(\boldsymbol{l}) + V(\boldsymbol{l})) \leq \sup_{\boldsymbol{l} \in \mathcal{D}} U(\boldsymbol{l}) + \sup_{\boldsymbol{l} \in \mathcal{D}} V(\boldsymbol{l}), \tag{39}$$

for any functions U and V, we obtain:

$$\begin{aligned}
&\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\mathcal{P}^*(\boldsymbol{l})-\rho\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2\\
&\le \underbrace{\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\mathcal{P}^*(\boldsymbol{l})-\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*(\boldsymbol{d}_k)\right)\right\|_2}_{=\epsilon(b_\phi)}\\
&\quad+\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*(\boldsymbol{d}_k)\right)-\rho\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2.
\end{aligned}$$

Step 2.3: The inequality above holds for any $\phi\in\mathcal{H}$ and $\rho\in\mathcal{V}$. Taking the infimum over all such functions on both sides preserves the inequality:

$$\begin{aligned}
\epsilon_{\mathrm{p}} &= \inf_{\substack{\phi\in\mathcal{H}\\ \rho\in\mathcal{V}}}\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\mathcal{P}^*(\boldsymbol{l})-\rho\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2\\
&\le \inf_{\substack{\phi\in\mathcal{H}\\ \rho\in\mathcal{V}}}\sup_{\boldsymbol{l}\in\mathcal{D}}\Bigg(\epsilon(b_\phi)\\
&\qquad+\left\|\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*(\boldsymbol{d}_k)\right)-\rho\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2\Bigg)\\
&= \epsilon(b_\phi)\\
&\quad+\underbrace{\inf_{\substack{\phi\in\mathcal{H}\\ \rho\in\mathcal{V}}}\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*(\boldsymbol{d}_k)\right)-\rho\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2}_{\triangleq\epsilon_1}.
\end{aligned}\tag{40}$$

Step 2.4: To bound ϵ_1, we add and subtract $\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*_{\mathrm{nn}}(\boldsymbol{d}_k)\right)$ and apply the triangle inequality:

$$\begin{aligned}
&\left\|\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*(\boldsymbol{d}_k)\right)-\rho\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2\\
&\le\left\|\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*(\boldsymbol{d}_k)\right)-\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*_{\mathrm{nn}}(\boldsymbol{d}_k)\right)\right\|_2\\
&\quad+\left\|\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*_{\mathrm{nn}}(\boldsymbol{d}_k)\right)-\rho\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2.
\end{aligned}$$

Step 2.5: Applying the supremum over $\boldsymbol{l}\in\mathcal{D}$ to both sides:

$$\begin{aligned}
&\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*(\boldsymbol{d}_k)\right)-\rho\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2\\
&\le\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*(\boldsymbol{d}_k)\right)-\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*_{\mathrm{nn}}(\boldsymbol{d}_k)\right)\right\|_2\\
&\quad+\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*_{\mathrm{nn}}(\boldsymbol{d}_k)\right)-\rho\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2.
\end{aligned}$$

Step 2.6: Using the Lipschitz continuity of ρ^* with constant $\mathrm{Lip}(\rho^*)$:

$$\begin{aligned}
&\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*(\boldsymbol{d}_k)\right)-\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*_{\mathrm{nn}}(\boldsymbol{d}_k)\right)\right\|_2\\
&\le\sup_{\boldsymbol{l}\in\mathcal{D}}\mathrm{Lip}(\rho^*)\left\|\sum_{k=1}^{K}\phi^*(\boldsymbol{d}_k)-\sum_{k=1}^{K}\phi^*_{\mathrm{nn}}(\boldsymbol{d}_k)\right\|_2\\
&\le\sup_{\boldsymbol{l}\in\mathcal{D}}\mathrm{Lip}(\rho^*)\sum_{k=1}^{K}\|\phi^*(\boldsymbol{d}_k)-\phi^*_{\mathrm{nn}}(\boldsymbol{d}_k)\|_2\\
&\le K\,\mathrm{Lip}(\rho^*)\epsilon_{\phi_{\mathrm{nn}}}.
\end{aligned}\tag{41}$$

Step 2.7: Combining the results from Steps 2.5 and 2.6:

$$\begin{aligned}
&\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*(\boldsymbol{d}_k)\right)-\rho\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2\\
&\le K\,\mathrm{Lip}(\rho^*)\epsilon_{\phi_{\mathrm{nn}}}\\
&\quad+\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*_{\mathrm{nn}}(\boldsymbol{d}_k)\right)-\rho\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2.
\end{aligned}$$

Step 2.8: Taking the infimum over all $\phi\in\mathcal{H}$ and $\rho\in\mathcal{V}$:

$$\begin{aligned}
\epsilon_1 &\le K\,\mathrm{Lip}(\rho^*)\epsilon_{\phi_{\mathrm{nn}}}\\
&\quad+\underbrace{\inf_{\substack{\phi\in\mathcal{H}\\ \rho\in\mathcal{V}}}\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*_{\mathrm{nn}}(\boldsymbol{d}_k)\right)-\rho\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2}_{\triangleq\epsilon_2}.
\end{aligned}$$

Step 2.9: For ϵ_2, since $\phi^*_{\mathrm{nn}}\in\mathcal{H}$, we have:

$$\begin{aligned}
&\inf_{\substack{\phi\in\mathcal{H}\\ \rho\in\mathcal{V}}}\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*_{\mathrm{nn}}(\boldsymbol{d}_k)\right)-\rho\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2\\
&\le\inf_{\rho\in\mathcal{V}}\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\rho^*\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*_{\mathrm{nn}}(\boldsymbol{d}_k)\right)-\rho\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*_{\mathrm{nn}}(\boldsymbol{d}_k)\right)\right\|_2.
\end{aligned}$$

Step 2.10: Since the vector $\left(\boldsymbol{d}_0,\sum_{k=1}^{K}\phi^*_{\mathrm{nn}}(\boldsymbol{d}_k)\right)\in\mathbb{R}^{m+b_\phi}=\mathcal{Y}$, we can bound ϵ_2 as:

$$\begin{aligned}
\epsilon_2 &\le \inf_{\rho\in\mathcal{V}}\sup_{\boldsymbol{y}\in\mathcal{Y}}\|\rho^*(\boldsymbol{y})-\rho(\boldsymbol{y})\|_2\\
&= \epsilon_{\rho_{\mathrm{nn}}}.
\end{aligned}$$

Step 2.11: Combining all results from Steps 2.3, 2.8, and 2.10:

$$\begin{aligned}
\epsilon_{\mathrm{p}} &\le \epsilon(b_\phi)+\epsilon_1\\
&\le \epsilon(b_\phi)+K\,\mathrm{Lip}(\rho^*)\epsilon_{\phi_{\mathrm{nn}}}+\epsilon_2.\\
&\le \epsilon(b_\phi)+K\,\mathrm{Lip}(\rho^*)\epsilon_{\phi_{\mathrm{nn}}}+\epsilon_{\rho_{\mathrm{nn}}}.
\end{aligned}\tag{42}$$

Conclusion: Combining the results from Parts 1 and 2, we have established that:

$$\epsilon(b_\phi)\le\epsilon_p\le\epsilon(b_\phi)+K\cdot\mathrm{Lip}(\rho^*)\epsilon_{\phi_{\mathrm{nn}}}+\epsilon_{\rho_{\mathrm{nn}}},\tag{43}$$

which completes the proof.

□

APPENDIX G
SECOND-STAGE SUB-PROBLEM

The second-stage sub-problem is formulated as follows:

$$\min \ \sum\nolimits_{i\in\mathcal{B}} c_{i1}\cdot(p_i^g)^2 + c_{i2}\cdot p_i^g + c_{i3} \tag{44}$$
$$\text{s.t. } p_{ij}^f = g_{ij}v_i^2 - v_iv_j(g_{ij}\cos\theta_{ij} + b_{ij}\sin\theta_{ij}), (i,j)\in\mathcal{E}, \tag{45}$$
$$q_{ij}^f = -b_{ij}v_i^2 - v_iv_j(g_{ij}\sin\theta_{ij} - b_{ij}\cos\theta_{ij}), (i,j)\in\mathcal{E}, \tag{46}$$
$$p_i^g - p_{ik}^d = \sum\nolimits_{(i,j)\in\mathcal{E}} p_{ij}^f, i\in\mathcal{B}, \tag{47}$$
$$q_i^g - q_{ik}^d = \sum\nolimits_{(i,j)\in\mathcal{E}} q_{ij}^f,\ i\in\mathcal{B}, \tag{48}$$
$$\underline{p_i^g} \le p_i^g \le \overline{p_i^g},\ i\in\mathcal{B}, \tag{49}$$
$$\underline{q_i^g} \le q_i^g \le \overline{q_i^g},\ i\in\mathcal{B}, \tag{50}$$
$$\underline{v_i} \le v_i \le \overline{v_i},\ i\in\mathcal{B}, \tag{51}$$
$$\underline{\theta_{ij}} \le \theta_{ij} = \theta_i - \theta_j \le \overline{\theta_{ij}}, (i,j)\in\mathcal{E}, \tag{52}$$
$$(p_{ij}^f)^2 + (q_{ij}^f)^2 \le (\overline{s_{ij}})^2, (i,j)\in\mathcal{E}, \tag{53}$$
$$|p_i^g - p_{i0}^{g*}| \le \Delta p_i,\ i\in\mathcal{B}, \tag{54}$$
$$\text{var. } p_i^g, q_i^g, \theta_i, v_i,\ i\in\mathcal{B}. \tag{55}$$

Here p_{i0}^{g*} is the given optimal first-stage active power generation at bus i. p_{ik}^d and q_{ik}^d are the active and reactive load at bus i in scenario k. p_i^g and q_i^g represent the active and reactive power generation at bus i. v_i and θ_i indicates the voltage magnitude and angle at bus i. p_{ij}^f and q_{ij}^f are the active and reactive power flow at line (i,j).

Constraints (45) and (46) define the active and reactive power flow. The power balance is ensured by constraints (47) and (48). Constraints (49) and (50) capture the active and reactive power generation limits. The voltage magnitude and phase angle limits are described in constraints (51) and (52). Constraint (53) ensures the branch flow limit. Constraint (54) ensures the ramping limits on active power generation. The objective is to minimize the second-stage generation cost.

APPENDIX H
CHANCE-CONSTRAINED AC-OPF

We use the following chance-constrained AC-OPF problem as our baseline:

$$\min \ \sum_{i\in\mathcal{B}} c_{i,1}\cdot(p_{i0}^g)^2 + c_{i,2}\cdot p_{i0}^g + c_{i,3} + \sum_{i\in\mathcal{B}} \mathbb{E}_{\boldsymbol{\xi}}\left[c_{i,1}\cdot(p_i^g(\boldsymbol{\xi}))^2 + c_{i,2}\cdot p_i^g(\boldsymbol{\xi}) + c_{i,3}\right] \tag{56}$$
$$\text{s.t. } p_{ij0}^f = g_{ij}v_{i0}^2 - v_{i0}v_{j0}(g_{ij}\cos\theta_{ij0} + b_{ij}\sin\theta_{ij0}), (i,j)\in\mathcal{E}, \tag{57}$$
$$q_{ij0}^f = -b_{ij}v_{i0}^2 - v_{i0}v_{j0}(g_{ij}\sin\theta_{ij0} - b_{ij}\cos\theta_{ij0}), (i,j)\in\mathcal{E}, \tag{58}$$
$$p_{i0}^g - p_{i0}^d = \sum\nolimits_{(i,j)\in\mathcal{E}} p_{ij0}^f,\ i\in\mathcal{B}, \tag{59}$$
$$q_{i0}^g - q_{i0}^d = \sum\nolimits_{(i,j)\in\mathcal{E}} q_{ij0}^f,\ i\in\mathcal{B}, \tag{60}$$
$$\underline{p_i^g} \le p_{i0}^g \le \overline{p_i^g},\ i\in\mathcal{B}, \tag{61}$$
$$\underline{q_i^g} \le q_{i0}^g \le \overline{q_i^g},\ i\in\mathcal{B}, \tag{62}$$
$$\underline{v_i} \le v_{i0} \le \overline{v_i},\ i\in\mathcal{B}, \tag{63}$$
$$\underline{\theta_{ij}} \le \theta_{ij0} = \theta_{i0} - \theta_{j0} \le \overline{\theta_{ij}},\ (i,j)\in\mathcal{E}, \tag{64}$$
$$(p_{ij0}^f)^2 + (q_{ij0}^f)^2 \le (\overline{s_{ij}})^2, (i,j)\in\mathcal{E}, \tag{65}$$
$$p_{ij}^f(\hat{\xi}) = g_{ij}v_i(\hat{\xi})^2 - v_i(\hat{\xi})v_j(\hat{\xi})(g_{ij}\cos\theta_{ij}(\hat{\xi}) + b_{ij}\sin\theta_{ij}(\hat{\xi})),\ (i,j)\in\mathcal{E},\ \hat{\xi}\in\Xi, \tag{66}$$
$$q_{ij}^f(\hat{\xi}) = -b_{ij}v_i(\hat{\xi})^2 - v_i(\hat{\xi})v_j(\hat{\xi})(g_{ij}\sin\theta_{ij}(\hat{\xi}) - b_{ij}\cos\theta_{ij}(\hat{\xi})),\ (i,j)\in\mathcal{E},\ \hat{\xi}\in\Xi, \tag{67}$$
$$p_i^g(\hat{\xi}) - p_i^d(\hat{\xi}) = \sum\nolimits_{(i,j)\in\mathcal{E}} p_{ij}^f(\hat{\xi}),\ i\in\mathcal{B},\ \hat{\xi}\in\Xi, \tag{68}$$
$$q_i^g(\hat{\xi}) - q_i^d(\hat{\xi}) = \sum\nolimits_{(i,j)\in\mathcal{E}} q_{ij}^f(\hat{\xi}),\ i\in\mathcal{B},\ \hat{\xi}\in\Xi, \tag{69}$$
$$\mathbb{P}_{\boldsymbol{\xi}}\left\{\begin{array}{l} \underline{p_i^g} \le p_i^g(\boldsymbol{\xi}) \le \overline{p_i^g}, i\in\mathcal{B}, \\ \underline{q_i^g} \le q_i^g(\boldsymbol{\xi}) \le \overline{q_i^g}, i\in\mathcal{B}, \\ \underline{v_i} \le v_i(\boldsymbol{\xi}) \le \overline{v_i}, i\in\mathcal{B}, \\ \underline{\theta_{i,j}} \le \theta_i(\boldsymbol{\xi}) - \theta_j(\boldsymbol{\xi}) \le \overline{\theta_{ij}}, (i,j)\in\mathcal{E}, \\ (p_{i,j}^f(\boldsymbol{\xi}))^2 + (q_{i,j}(\boldsymbol{\xi})^f)^2 \le (\overline{s_{i,j}})^2, (i,j)\in\mathcal{E}, \\ |p_i^g(0) - p_i^g(\boldsymbol{\xi})| \le \Delta p_i, i\in\mathcal{B}, \end{array}\right\} \ge \delta. \tag{70}$$
$$\text{var. } p_{i0}^g, q_{i0}^g, \theta_{i0}, v_{i0}, p_i^g(\hat{\xi}), q_i^g(\hat{\xi}), \theta_i(\hat{\xi}), v_i(\hat{\xi}),\ i\in\mathcal{B}, \hat{\xi}\in\Xi. \tag{71}$$

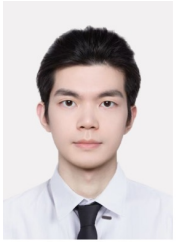
Here, $\boldsymbol{\xi}$ denotes the random load in the second stage, with support set Ξ. $\hat{\xi}$ is a realization of $\boldsymbol{\xi}$. $p_i^d(\hat{\xi})$ and $q_i^d(\hat{\xi})$ are the active and reactive load at bus i under demand $\hat{\xi}$. $p_i^g(\hat{\xi})$ and $q_i^g(\hat{\xi})$ represents the active and reactive power generation at bus i under load realization $\hat{\xi}$. $v_i(\hat{\xi})$ and $\theta_i(\hat{\xi})$ indicates the voltage magnitude and angle at bus i under realization $\hat{\xi}$. $p_{ij}^f(\hat{\xi})$ and $q_{ij}^f(\hat{\xi})$ are the active and reactive power flow at line (i,j) under realization $\hat{\xi}$.

Constraints (57) and (58) define the active and reactive power flow in the first stage. The power balance in the first stage is ensured by constraints (59) and (60). Constraints (61) and (62) capture the active and reactive power generation limits in the first stage. The voltage magnitude and phase angle limits in the first stage are described in constraints (63) and (64). Constraint (65) ensures the branch flow limit in the first stage. Constraints (66) and (67) define the active and reactive power flow for a given load realization $\hat{\xi}$ in the second stage. The power balance in the second stage is ensured by constraints (68) and (69). Constraint (70) ensures the probability of inequality constraint satisfaction does not is

greater than a predefined confidence level δ. In our simulation, δ is set as 99% and 95% for the 118-bus and 793-bus test systems, respectively. By allowing for a predefined probability of constraint violation, this chance-constrained AC-OPF formulation achieves a balance between solution robustness and cost-effectiveness.

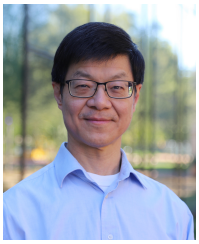
We use the data-driven method in [7] to solve this challenging chance-constrained AC-OPF problem. Instead of using random scenario sampling, this method designed an iterative scenario selection strategy to identify and add critical scenarios to the scenario-based AC-OPF formulation, until a predefined confidence level is achieved. See [7] for more details on this data-driven method.

Min Zhou received the B.Eng. degree in 2017 and the M.S. degree in 2020, both from the School of Management of Engineering, Nanjing University, Nanjing, China. He is currently pursuing the Ph.D. degree with the Department of Data Science, City University of Hong Kong, Hong Kong, where he is a recipient of the City University of Hong Kong Presidential PhD Scholarship. His research interests include machine learning and its applications to power system analysis and optimization.



Enming Liang received the B.Eng. degree in intelligent systems engineering from Sun Yat-sen University, Guangzhou, China, in 2020, and the Ph.D. degree in data science from City University of Hong Kong, Hong Kong, in 2024. He is currently a Research Assistant Professor with the College of Computing, City University of Hong Kong. His research interests lie at the intersection of machine learning and constrained optimization, with a focus on their applications in power systems, mobility networks, and climate resilience. Dr. Liang has published extensively in leading venues including JMLR, IEEE TNNLS, ICML, ICLR, NeurIPS, and AAAI. His work has been recognized with the Outstanding Short Paper Award at the ICLR 2025 DeLTa workshop and the Outstanding Academic Performance Award from City University of Hong Kong in 2023. He also received the Excellence Award in the Star of Tomorrow Internship Program from Microsoft Research Asia in 2022 and achieved second place in the ACM KDD Cup competition in 2020. Dr. Liang serves as a reviewer for premier machine learning conferences including NeurIPS, ICML, and ICLR, and was honored with the Top Reviewer Award at NeurIPS 2024.

Minghua Chen (S'04 M'06 SM' 13 F'22) received his B.Eng. and M.S. degrees from the Department of Electronic Engineering at Tsinghua University. He received his Ph.D. degree from the Department of Electrical Engineering and Computer Sciences at University of California Berkeley. He is currently a Presidential Chair Professor at School of Data Science, The Chinese University of Hong Kong, Shenzhen. He is on leave with Department of Data Science, City University of Hong Kong. Minghua received the Eli Jury award from UC Berkeley in 2007 (presented to a graduate student or recent alumnus for outstanding achievement in the area of Systems, Communications, Control, or Signal Processing) and The Chinese University of Hong Kong Young Researcher Award in 2013. He also received IEEE ICME Best Paper Award in 2009, IEEE Transactions on Multimedia Prize Paper Award in 2009, ACM Multimedia Best Paper Award in 2012, IEEE INFOCOM Best Poster Award in 2021, ACM e-Energy Best Paper Award in 2023, and Gradient AI Research Award in 2024. He receives the ACM Recognition of Service Award in 2017 and 2020 for the service contribution to the research community. He is currently a Senior Editor for IEEE Systems Journal and Award Chair of ACM SIGEnergy. Minghua's recent research interests include online optimization and algorithms, machine learning with hard constraints and its applications in power systems, intelligent transportation systems, distributed optimization, and delay-critical networked systems. He is an ACM Distinguished Scientist and IEEE Fellow.



Steven Low is the F. J. Gilloon Professor of the Computing & Mathematical Sciences Department and Electrical Engineering Department at Caltech. Before that, he was with AT&T Bell Laboratories, Murray Hill, NJ, and the University of Melbourne, Australia. He has held honorary/chaired professorship in Australia, China and Taiwan. He was a co-recipient of IEEE best paper awards, an awardee of the IEEE Koji Kobayashi Computers and Communications Award, the IEEE INFOCOM Achievement Award and the ACM SIGMETRICS Test of Time Award, and is a Fellow of IEEE, ACM, and CSEE. He is well-known for work on Internet congestion control and optimal power flow problems in smart grid. His research has been deployed on the Internet for content distribution since 2012 and in the US for large-scale workplace electric vehicle charging since 2019. He is the author of Power System Analysis: Analytical tools and structural properties, to be published by Cambridge. He received his B.S. from Cornell and PhD from Berkeley, both in EE.