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

(Technical Report, DANCE Systems Lab, City University of Hong Kong, December 2024)

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) a novel partially permutation-invariant neural network (PPNN) 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, comparable second-stage out-of-sample feasibility, with 0.82% optimality loss.**

***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^d_{ik}, q^d_{ik}	Active and reactive load at bus i, scenario k
p^f_{ijk}	Active power flow at line (i, j), scenario k
q^f_{ijk}	Reactive power flow at line (i, j), scenario k
v_{ik}, θ_{ik}	Voltage magnitude and angle at bus i, scenario k
p^g_{ik}, q^g_{ik}	Active and reactive generation at bus i, scenario k

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, 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

M. Zhou, E. Liang, and M. Chen are with Department of Data Science, City University of Hong Kong. S. Low is with Computing and Mathematical Sciences and Electrical Engineering, Caltech. Corresponding author: M. Chen.

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

Category	Approach	Related work	OPF model	ML model	Metrics in consideration		
					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.

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 a novel PPNN architecture 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 an optimality gap of merely 0.82% 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 practical usefulness in capturing complex spatial and temporal correlations in renewable and load inputs [8], [9]. Along this line, 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 stochastic OPF problems. 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. A framework for tackling two-stage stochastic problem is proposed in [58], 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 two-stage stochastic AC-OPF problems 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* 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 [59]:

$$
\begin{aligned}
\text{min}\ & \sum\nolimits_{i\in\mathcal{B}} \left[c_{i1}\cdot (p^g_{i0})^2 + c_{i2}\cdot p^g_{i0} + c_{i3}\right] \\
& + \frac{1}{K}\sum\nolimits_{k=1}^{K}\sum\nolimits_{i\in\mathcal{B}} \left[c_{i1}\cdot (p^g_{ik})^2 + c_{i2}\cdot p^g_{ik} + c_{i3}\right] && (1)\\
\text{s.t.}\ & p^f_{ijk} = g_{ij}v_{ik}^2 - v_{ik}v_{jk}(g_{ij}\cos\theta_{ijk} + b_{ij}\sin\theta_{ijk}), \\
& (i,j)\in\mathcal{E},\ k=0,\dots,K, && (2)\\
& q^f_{ijk} = -b_{ij}v_{ik}^2 - v_{ik}v_{jk}(g_{ij}\sin\theta_{ijk} - b_{ij}\cos\theta_{ijk}), \\
& (i,j)\in\mathcal{E},\ k=0,\dots,K, && (3)\\
& p^g_{ik} - p^d_{ik} = \sum\nolimits_{(i,j)\in\mathcal{E}} p^f_{ijk},\ i\in\mathcal{B},\ k=0,\dots,K, && (4)\\
& q^g_{ik} - q^d_{ik} = \sum\nolimits_{(i,j)\in\mathcal{E}} q^f_{ijk},\ i\in\mathcal{B},\ k=0,\dots,K, && (5)\\
& \underline{p^g_i} \le p^g_{ik} \le \overline{p^g_i},\ i\in\mathcal{B},\ k=0,\dots,K, && (6)\\
& \underline{q^g_i} \le q^g_{ik} \le \overline{q^g_i},\ i\in\mathcal{B},\ k=0,\dots,K, && (7)\\
& \underline{v_i} \le v_{ik} \le \overline{v_i},\ i\in\mathcal{B},\ k=0,\dots,K, && (8)\\
& \underline{\theta_{ij}} \le \theta_{ijk} = \theta_{ik} - \theta_{jk} \le \overline{\theta_{ij}}, (i,j)\in\mathcal{E}, k=0,\dots,K, && (9)\\
& (p^f_{ijk})^2 + (q^f_{ijk})^2 \le (\overline{s_{ij}})^2, (i,j)\in\mathcal{E},\ k=0,\dots,K, && (10)\\
& |p^g_{ik} - p^g_{i0}| \le \Delta p_i,\ i\in\mathcal{B},\ k=1,\dots,K, && (11)\\
\text{var.}\ & p_{ik}, q_{ik}, \theta_{ik}, v_{ik},\ i\in\mathcal{B},\ \ k=0,\dots,K. && (12)
\end{aligned}
$$

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. Equations (2) and (3) define the active and reactive power flow at each scenario. The power balance at each scenario is ensured by Equations (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 defined as a bijection π from the index set to itself. We use π_i to denote the ith element of the permuted index set. 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 permutation of the index set $\{1,2,3\}$ is $\pi=\{1\to 2, 2\to 3, 3\to 1\}$, with permuted indices as $\{\pi_1=2, \pi_2=3, \text{and } \pi_3=1\}$. For a vector $(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3)$, the corresponding permuted vector is $\pi(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3)=(\boldsymbol{x}_2,\boldsymbol{x}_3,\boldsymbol{x}_1)$.

We now analyze the load to first-stage solution mapping, denoted as $\mathcal{P}^*$, in the two-stage stochastic AC-OPF problem. For brevity, let $\boldsymbol{d}_0$ denote the first-stage net load, and $\boldsymbol{d}_k$ represent the net load inputs in the k-th second-stage scenario.

Theorem 2. *(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,*

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

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

See Appendix A for the proof of Theorem 2. Theorem 2 identifies an elegant partial permutation-invariance property in the two-stage stochastic AC-OPF problem. Intuitively, this property comes from the observation that, when solving the two-stage stochastic AC-OPF problem, switching the order of the second-stage scenarios does not change the first-stage solution[1]. Next, we establish an important structural expression of the mapping $\mathcal{P}^*$, by applying the Kolmogorov-Arnold theorem [60] to the scenario-based two-stage stochastic AC-OPF problem setting.

Theorem 3. *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 3 is proved in Appendix C. This decomposition suggests that we can learn two mappings, ϕ and ρ, with dimensions independent of the number of scenarios K, to represent the target mapping $\mathcal{P}^*$, without suffering from the dimensionality explosion issue.

While Theorem 3 provides an upper bound of K^5m^2 for the embedding dimension b_ϕ (i.e., the output dimension of ϕ), this bound may not be tight for specific problems [61]. Determining the minimal b_ϕ for a specific partially permutation-invariant mapping remains largely open. In this study, we follow an empirical approach common in the literature [61], [62] to decide b_ϕ by incrementally increasing its value until a satisfactory empirical performance is achieved. We observe that setting b_ϕ to a value similar to the dimension of the load input in each scenario m, independent of the number of scenarios K, yields satisfactory performance empirically.

[1]We discuss intuitively why exploiting this partial permutation-invariance property would enable more efficient machine learning schemes in Appendix G.

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}	$\theta_{ik}, v_{ik}, i\in\mathcal{B}$	$\theta_{ik}, q^g_{ik}, i\in\mathcal{G}$

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 a 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 neural network 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.

B. The *DeepOPF-Stoc* architecture

We exploit the partial permutation-invariance property of structural representation in Theorem 3 to design a PPNN model for learning the input to the first-stage solution mapping in the two-stage stochastic AC-OPF problem efficiently. The PPNN architecture is shown in Figure 1. Instead of concatenating the load inputs into a high-dimensional vector, PPNN first

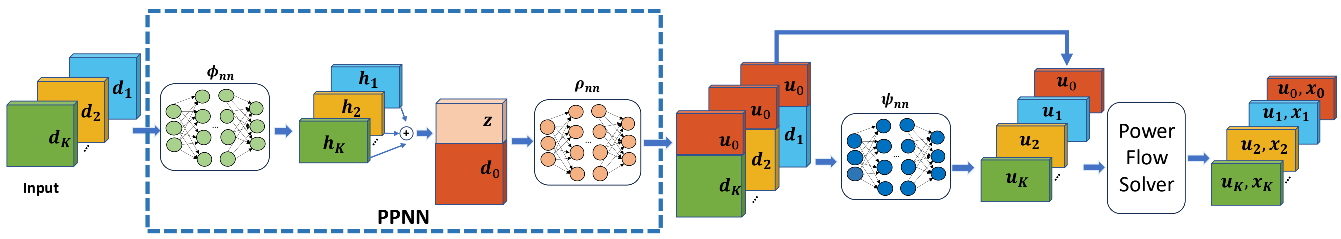

Fig. 1. Schema of DeepOPF-Stoc. First, the inputs in each second-stage scenario are fed separately into a neural network, denoted as ϕ_{nn}, to generate their hidden representations. Next, these representations are summed to create an embedding $\boldsymbol{z}$, which is then concatenated with the first-stage 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 $\boldsymbol{u}_0$ is then concatenated with the inputs in each second-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.

uses a neural network, denoted as ϕ_{nn}, to process the inputs of each second-stage scenario separately:

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

where $\boldsymbol{d}_k$ is the load inputs of the k-th scenario. ϕ_{nn} is a FCNN with ReLU activation functions to approximate the continuous mapping ϕ in Theorem 3. 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:

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

where ρ_{nn} is also a FCNN with ReLU activation functions to approximate the continuous mapping ρ in Theorem 3.

Similar to [58], to get the second-stage solutions, we train one DNN to solve multiple second-stage sub-problems by learning a mapping $\psi_{\mathrm{nn}}(\cdot)$ from any first-stage solution $\boldsymbol{u}_0$ and second-stage input $\boldsymbol{d}_k$ to the second-stage solution,

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

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

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

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.

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 3. 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.

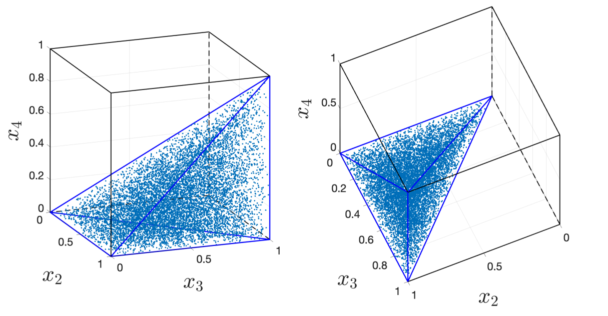

Fig. 2. A three-dimensional visualization of the samples obtained by the PPNN-aware sampling algorithm. The target mapping is $\mathcal{P}(\boldsymbol{x}) = x_1 + \frac{1}{2}(x_2 + x_3 + x_4)$, with the input domain set as $[0,1]^4$. To eliminate sample redundancy, we employ the PPNN-aware sampling algorithm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

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 $\boldsymbol{l}_2 \in \mathcal{N}(\boldsymbol{d}_0, \pi(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K))$, 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:

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

Here, $f_{\boldsymbol{D}_0,\ldots,\boldsymbol{D}_K}(\boldsymbol{d}_0, \ldots, \boldsymbol{d}_K)$ 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 $f_{\boldsymbol{D}_0,\ldots,\boldsymbol{D}_K}(\boldsymbol{d}_0, \ldots, \boldsymbol{d}_K)$[2], we focus

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

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:

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

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 Equations (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 [63]. The gradient of $\mathcal{L}$ with respect to the parameters of ψ_{nn}, denoted as θ_ψ, is given by:

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

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 Equation (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 Equation (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 Equations (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 Equations (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 4. 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

$$\epsilon(b_\phi) \triangleq \inf_{\substack{\phi\in\mathcal{F}\\ \rho\in\mathcal{Q}}} \sup_{\boldsymbol{l}\in\mathcal{D}} \left\|\mathcal{P}^*(\boldsymbol{l}) - \rho\left(\boldsymbol{d}_0, \sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2^2, \tag{19}$$

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 3 in Section III-B, the error $\epsilon(b_\phi)$ in Equation (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 5. *For any continuous mappings ρ^* and ϕ^* that achieve the minimal $\epsilon(b_\phi)$ in Equation (19), there exists a PPNN composed of neural networks ρ_{nn} and ϕ_{nn}, such that its approximation error to $\mathcal{P}^*$, defined as*

$$\epsilon_p \triangleq \inf_{\substack{\phi\in\mathcal{H}\\ \rho\in\mathcal{V}}} \sup_{\boldsymbol{l}\in\mathcal{D}} \left\|\mathcal{P}^*(\boldsymbol{l}) - \rho\left(\boldsymbol{d}_0, \sum_{k=1}^{K}\phi(\boldsymbol{d}_k)\right)\right\|_2^2,$$

is bounded by

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

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

$$\epsilon_{\phi_{\text{nn}}} \triangleq \inf_{\phi\in\mathcal{H}} \sup_{\boldsymbol{x}\in\mathcal{X}} \|\phi^*(\boldsymbol{x}) - \phi(\boldsymbol{x})\|_2^2,$$

$$\epsilon_{\rho_{\text{nn}}} \triangleq \inf_{\rho\in\mathcal{V}} \sup_{\boldsymbol{y}\in\mathcal{Y}} \|\rho^*(\boldsymbol{y}) - \rho(\boldsymbol{y})\|_2^2,$$

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

Theorem 5, proved in Appendix C., bounds the approximation error of PPNN to the target mapping $\mathcal{P}^*$. As ρ_{nn} and ϕ_{nn} are both FCNNs with ReLU activation functions as described in Section IV-B, they are universal approximators of any continuous mappings, including ρ^* and ϕ^*. Hence, with perfect training, the approximation errors $\epsilon_{\phi_{\text{nn}}}$ and $\epsilon_{\rho_{\text{nn}}}$ decreases to zero as the number of tunable parameters in ρ_{nn} and ϕ_{nn} increases, and the PPNN approximation error ϵ_{p} reaches its lower bound $\epsilon(b_\phi)$ as long as the number of scenarios K and the Lipschitz constant of ρ^* are finite. This theoretical foundation validates our PPNN architecture presented in Section IV. Furthermore, the error bound established in Theorem 5 reveals that the overall error scales linearly with the number of second-stage scenarios K, representing a substantial improvement over the vanilla architecture, as demonstrated in the numerical results in Section VI-C. The error expression further suggests that employing a large-scale neural network for ϕ_{nn} can help keep the overall approximation error small, particularly in cases involving large K values.

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 [64]. To simulate renewable generation, we modify the test system by adding negative load demand in selected buses (See our test cases in [65]). 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[3]. For each test case, we create 10,000 instances for training and 2,500 for testing. We implement DeepOPF-Stoc based on the PyTorch platform, with hyperparameters summarized in Table III. To ensure solution feasibility, we further integrate the bisection projection approach [55] into our DeepOPF-Stoc design, which ensures solution feasibility by using a trained feasibility-ensuring network to provide an initial feasible solution (See [55] for details). The load and renewable uncertainties are modeled as follows:

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, 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.

[3]We note that the scenario-based formulation transforms the original two-stage stochastic AC-OPF problem into a deterministic one, solvable using the MIPS solver. See Appendix F for the solution technique of MIPS.

TABLE III
DNN PARAMETER SETTINGS.

# Bus	# hidden layers[1]	# hidden neurons	Batch size	Training epoch	Learning rate	w_1	w_2	w_3	b_ϕ
118	2	128	32	20,000	1e-3	0.1	0.5	1	10
793	2	512	16	5,000	1e-3	0.1	2	1	30

[1]We use the same number of hidden layers/neurons for ϕ_{nn}, ρ_{nn} and ψ_{nn}.

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]; (ii) the chance-constrained AC-OPF approach (introduced in detail in Appendix D); (iii) The deterministic model, which simplifies the scenario-based approach by considering the future uncertainty as a deterministic vector. All baseline methods are implemented with Python.

Evaluation metrics: We evaluate the performance of each method using an out-of-sample analysis following standard practice for scenario-based approaches [23], [66]. For each first-stage solution obtained for IEEE 118-bus and synthetic 793-bus test systems, we randomly generate 200 out-of-sample scenarios from the in-sample distribution. For each first-stage solution and second-stage scenario, the second-stage subproblem (Appendix E) is then solved using a standard MIPS solver [2], one of the state-of-the-arts for solving non-convex optimization problems, to get the corresponding second-stage solutions. We consider the following evaluation metrics: (i) **Total cost**: the total cost of first-stage generation and the expected second-stage generation cost, where the latter is calculated by averaging the costs of feasible second-stage solutions across the out-of-sample scenarios. (ii) **Out-of-sample feasibility rate** ($\boldsymbol{\eta_{oos}}$): the proportion of out-of-sample scenarios for which a feasible second-stage solution exists. (iii) **Speedup** ($\boldsymbol{\eta_{sp}}$): the speedup achieved compared to MIPS.

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.

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 an optimality gap within 0.82%. 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 with a slightly higher total cost, 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

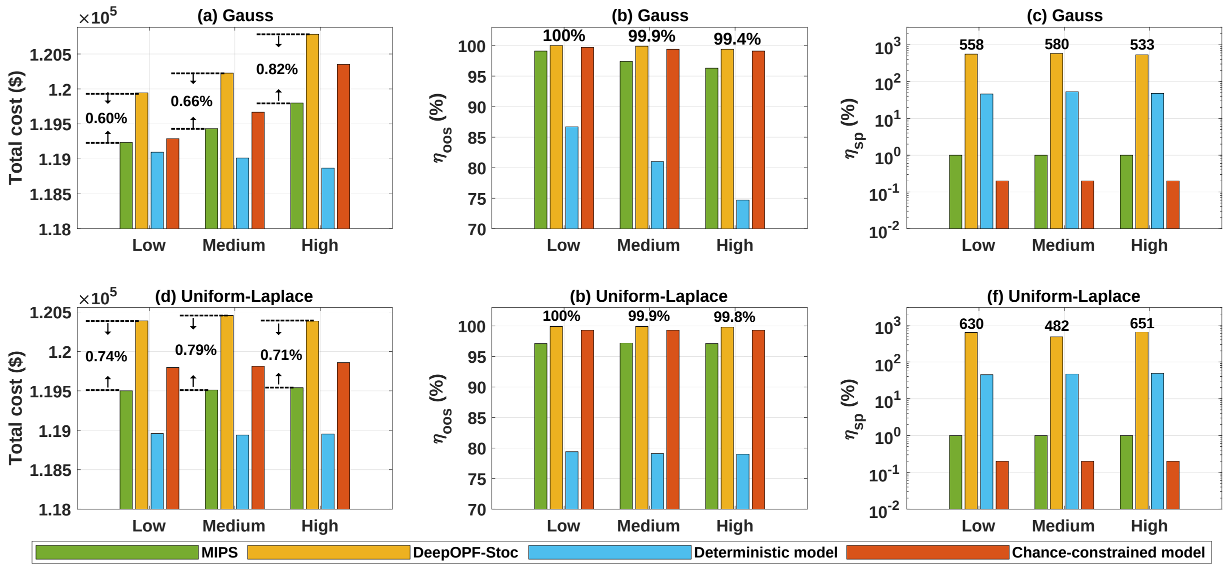

Fig. 3. Performance comparison of various methods under diverse uncertainty levels and distributions on the modified IEEE 118-bus test system.

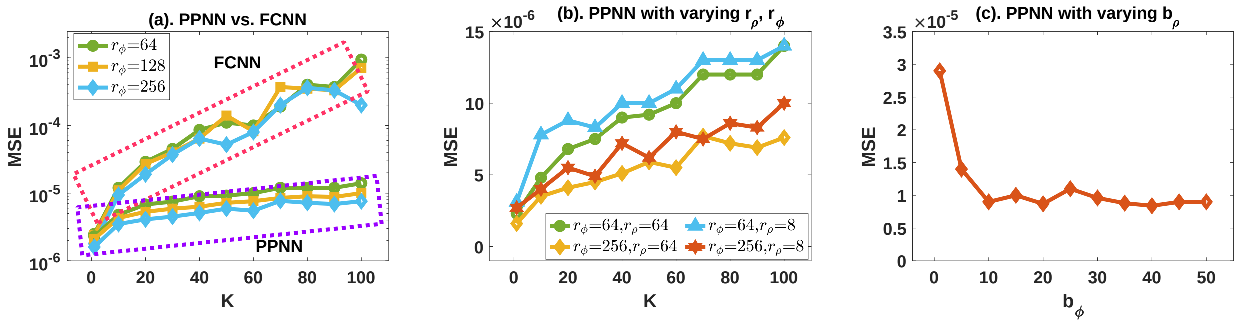

Fig. 4. MSE comparison of different methods across varying in-sample/network sizes.

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 total costs, as higher uncertainty levels either require increased reserves or lead to lower out-of-sample feasibility rates.

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 $\phi_{\rm nn}$ and $\rho_{\rm nn}$ 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 a FCNN. Using $r_\rho = 64$ and varying $r_\phi \in \{64, 128, 256\}$, we evaluate MSE of PPNN across different $\phi_{\rm nn}$ sizes. For each PPNN configuration, the FCNN is designed with a comparable number of tunable parameters. 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 $\rho_{\rm nn}$ 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 5, where the approximation errors of $\phi_{\rm nn}$ and $\rho_{\rm nn}$ determine the slope and vertical position of the curve[4]., 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,

[4]Our theoretical analysis employs the maximal squared error to measure approximation error of PPNN. In the simulation, we observed similar behaviors for the mean squared error, e.g., as K varies

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, in which the variance of the Gaussian distribution is 10% of the corresponding mean, described in Section VI-A.

Simulation results in Table IV show DeepOPF-Stoc generates feasible first-stage solutions within an optimality gap of 0.55% and a high out-of-sample feasibility rate, while achieving up to 250x speedup compared to MIPS. This highlights the scalability of our DeepOPF-Stoc design. 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 total costs and out-of-sample feasibility rate while being up to three orders of magnitude faster. The results also show that the out-of-sample feasibility rate of MIPS improves with the number of in-sample scenarios, demonstrating the benefit of using a larger scenario set in the scenario-based two-stage stochastic AC-OPF problem. Furthermore, the speedup achieved by DeepOPF-Stoc increases with the number of in-sample scenarios, further demonstrating its scalability.

Table IV also shows that DeepOPF-Stoc achieves a higher out-of-sample feasibility rate than MIPS. This can be attributed to their different mechanisms for ensuring second-stage feasibility. MIPS optimizes the first-stage solution by requiring only the existence of feasible second-stage solutions for in-sample scenarios, which can be found using primal-dual interior point methods. In contrast, DeepOPF-Stoc relies on a second-stage DNN to identify these solutions. Due to inherent prediction errors in the DNN, a larger feasible region is necessary to accommodate potential deviations. Consequently, DeepOPF-Stoc generates more conservative first-stage solutions than MIPS, resulting in enhanced out-of-sample feasibility rates.

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 an optimality gap within 0.82% while providing up to two orders of magnitude speedup compared to iterative methods.

TABLE IV
Simulation results on the 793-bus test system.

Methods	Metrics	size=5	size=10	size=15
MIPS	Total cost ($)	439,216	439,213	439,210
	η_{oos} (%)	93.7	95.5	96.4
Deterministic Model	Total cost ($)	439,228	439,228	439,229
	η_{oos} (%)	84.2	85.5	86.6
	η_{sp}	$\times 8$	$\times 29$	$\times 61$
Chance-constrained AC-OPF	Total cost ($)	439,225	439,226	439,227
	η_{oos} (%)	98.8	98.8	98.8
	η_{sp}	$\times 0.1$	$\times 0.2$	$\times 0.5$
DeepOPF-Stoc	Total cost ($)	440,117	440,799	441,626
	η_{oos} (%)	99.8	99.9	99.9
	η_{sp}	$\times 93$	$\times$ 169	$\times 250$

Beyond the two-stage stochastic AC-OPF problem, our DeepOPF-Stoc framework is also applicable to solving two-stage stochastic DC-OPF problems. Future research directions include (i) developing unsupervised learning approaches for two-stage stochastic AC-OPF problems, and (ii) adapting the proposed approach to two-stage stochastic AC-OPF problems with scenario aggregation/selection. These extensions would enable the incorporation of a large number of second-stage scenarios without requiring extensive labeled training data, thereby enhancing the framework's practical applicability.

Acknowledgement

GPT-based large language models developed by OpenAI were used in the paper for linguistic refinement.

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.

[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] "Efficient bisection projection to ensure nn solution feasibility for constrained optimization over non-convex set," 2024, submitted to *Proc. ICLR*. [Online]. Available: https://openreview.net/forum?id=7TXdglI1g0

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

[59] 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.

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

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

[62] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. IEEE CVPR*, 2017, pp. 652–660.

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

[64] 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.

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

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

[67] 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
Proof of Theorem 2

Proof. For notational convenience, we first write the scenario-based two-stage stochastic AC-OPF problem in the following compact form:

$$\min \; F\left(\boldsymbol{\zeta}_0\right)+\frac{1}{K}\sum_{k\in\mathcal{S}}F\left(\boldsymbol{\zeta}_k\right) \tag{20}$$

$$\text{s.t. } G\left(\boldsymbol{\zeta}_k,\boldsymbol{d}_k\right)=0,\;\; \forall k\in\{0\}\cup\mathcal{S}, \tag{21}$$

$$H\left(\boldsymbol{\zeta}_k,\boldsymbol{d}_k\right)\leq 0,\;\; \forall k\in\{0\}\cup\mathcal{S}, \tag{22}$$

$$|\boldsymbol{\zeta}_k-\boldsymbol{\zeta}_0|\leq\Delta,\;\; \forall k\in\mathcal{S}, \tag{23}$$

$$\text{var. } \boldsymbol{\zeta}_k,\;\; \forall k\in\{0\}\cup\mathcal{S}. \tag{24}$$

where $\boldsymbol{\zeta}_k$ denotes the decision variables, including the active and reactive power generation, voltage magnitude and angle, at scenario k. $F(\cdot)$ is the quadratic active power generation cost function defined in Equation (1). Constraint (21) defined the power balance equation and constraint (22) represents the inequality constraint for the decision variables at each scenario. Constraint (23) indicates the ramping constraint of the active power generation across two time periods.

For load input $(\boldsymbol{d}_0,\boldsymbol{d}_1,\cdots,\boldsymbol{d}_K)$, we denote the corresponding optimal solution as $(\boldsymbol{\zeta}_0^*,\boldsymbol{\zeta}_1^*,\cdots,\boldsymbol{\zeta}_K^*)$. We define the permutation π as a bijection from $\{1,2,\cdots,K\}$ to itself. The load input under permutation π is indicated as $\pi(\boldsymbol{d}_0,\boldsymbol{d}_1,\cdots,\boldsymbol{d}_K)=(\boldsymbol{d}_0,\boldsymbol{d}_{\pi_1},\cdots,\boldsymbol{d}_{\pi_K})$, and the optimal solution under permutation π is denoted as $\pi(\boldsymbol{\zeta}_0^*,\boldsymbol{\zeta}_1^*,\cdots,\boldsymbol{\zeta}_K^*)=(\boldsymbol{\zeta}_0^*,\boldsymbol{\zeta}_{\pi_1}^*,\cdots,\boldsymbol{\zeta}_{\pi_K}^*)$.

We will prove Theorem 2 by showing that $(\boldsymbol{\zeta}_0^*,\boldsymbol{\zeta}_{\pi_1}^*,\cdots,\boldsymbol{\zeta}_{\pi_K}^*)$ is the optimal solution of the scenario-based two-stage stochastic AC-OPF with load input $\pi(\boldsymbol{d}_0,\boldsymbol{d}_1,\cdots,\boldsymbol{d}_K)$. Thus, the first-stage optimal solution will remain unchanged under any permutation π applied to the second-stage scenarios.

We begin with showing $(\boldsymbol{\zeta}_0^*,\boldsymbol{\zeta}_{\pi_1}^*,\cdots,\boldsymbol{\zeta}_{\pi_K}^*)$ is a feasible solution to the two-stage stochastic AC-OPF problem with input $\pi(\boldsymbol{d}_0,\boldsymbol{d}_1,\cdots,\boldsymbol{d}_K)$. As $(\boldsymbol{\zeta}_0^*,\boldsymbol{\zeta}_1^*,\cdots,\boldsymbol{\zeta}_K^*)$ is the optimal solution of the two-stage stochastic AC-OPF problem with load input $(\boldsymbol{d}_0,\boldsymbol{d}_1,\cdots,\boldsymbol{d}_K)$, we have:

$$G\left(\boldsymbol{\zeta}_0^*,\boldsymbol{d}_0\right)=0,\;\; H\left(\boldsymbol{\zeta}_0^*,\boldsymbol{d}_0\right)\leq 0, \tag{25}$$

$$G\left(\boldsymbol{\zeta}_k^*,\boldsymbol{d}_k\right)=0,\;\; H\left(\boldsymbol{\zeta}_k^*,\boldsymbol{d}_k\right)\leq 0,\;\; \forall k\in\mathcal{S}, \tag{26}$$

$$|\boldsymbol{\zeta}_k^*-\boldsymbol{\zeta}_0^*|\leq\Delta,\;\; \forall k\in\mathcal{S}. \tag{27}$$

As π is a bijective mapping from $\{1,2,\cdots,K\}$ to itself, we can show that the following constraints are satisfied:

$$G\left(\boldsymbol{\zeta}_0^*,\boldsymbol{d}_0\right)=0,\;\; H\left(\boldsymbol{\zeta}_0^*,\boldsymbol{d}_0\right)\leq 0, \tag{28}$$

$$G\left(\boldsymbol{\zeta}_{\pi_k}^*,\boldsymbol{d}_{\pi_k}\right)=0,\;\; H\left(\boldsymbol{\zeta}_{\pi_k}^*,\boldsymbol{d}_{\pi_k}\right)\leq 0,\;\; \forall k\in\mathcal{S}, \tag{29}$$

$$\left|\boldsymbol{\zeta}_{\pi_k}^*-\boldsymbol{\zeta}_0^*\right|\leq\Delta,\;\; \forall k\in\mathcal{S}. \tag{30}$$

Thus, we can show that $(\boldsymbol{\zeta}_0^*,\boldsymbol{\zeta}_{\pi_1}^*,\cdots,\boldsymbol{\zeta}_{\pi_K}^*)$ is a **feasible** solution to the stochastic AC-OPF problem with load input $(\boldsymbol{d}_0,\boldsymbol{d}_{\pi_1},\cdots,\boldsymbol{d}_{\pi_K})$. We then prove the **optimality** of $(\boldsymbol{\zeta}_0^*,\boldsymbol{\zeta}_{\pi_1}^*,\cdots,\boldsymbol{\zeta}_{\pi_K}^*)$ by contradiction. Suppose $(\boldsymbol{\zeta}_0^*,\boldsymbol{\zeta}_{\pi_1}^*,\cdots,\boldsymbol{\zeta}_{\pi_K}^*)$ is not the optimal solution to the two-stage stochastic AC-OPF problem with load input $(\boldsymbol{d}_0,\boldsymbol{d}_{\pi_1},\cdots,\boldsymbol{d}_{\pi_K})$, we denote the associated optimal solution as $(\boldsymbol{\zeta}_0',\boldsymbol{\zeta}_{\pi_1}',\cdots,\boldsymbol{\zeta}_{\pi_K}')$. Similarly, we can show that $(\boldsymbol{\zeta}_0',\boldsymbol{\zeta}_1',\cdots,\boldsymbol{\zeta}_K')$ is a feasible solution to the two-stage stochastic AC-OPF problem with input $(\boldsymbol{d}_0,\boldsymbol{d}_1,\cdots,\boldsymbol{d}_K)$. Moreover, as $(\boldsymbol{\zeta}_0',\boldsymbol{\zeta}_{\pi_1}',\cdots,\boldsymbol{\zeta}_{\pi_K}')$ is the only optimal solution for the two-stage stochastic AC-OPF problem with input $(\boldsymbol{d}_0,\boldsymbol{d}_{\pi_1},\cdots,\boldsymbol{d}_{\pi_K})$, the following inequality holds:

$$F\left(\boldsymbol{\zeta}_0'\right)+\frac{1}{K}\sum_{k\in\mathcal{S}}F\left(\boldsymbol{\zeta}_{\pi_k}'\right)<F\left(\boldsymbol{\zeta}_0^*\right)+\frac{1}{K}\sum_{k\in\mathcal{S}}F\left(\boldsymbol{\zeta}_{\pi_k}^*\right). \tag{31}$$

Given that π is a bijective mapping from $\{1,2,\cdots,K\}$ to itself, we can rewrite this equation as:

$$F\left(\boldsymbol{\zeta}_0'\right)+\frac{1}{K}\sum_{k\in\mathcal{S}}F\left(\boldsymbol{\zeta}_k'\right)<F\left(\boldsymbol{\zeta}_0^*\right)+\frac{1}{K}\sum_{k\in\mathcal{S}}F\left(\boldsymbol{\zeta}_k^*\right). \tag{32}$$

This contradicts the assumption that $(\boldsymbol{\zeta}_0^*,\boldsymbol{\zeta}_1^*,\cdots,\boldsymbol{\zeta}_K^*)$ is the unique optimal solution to the two-stage stochastic AC-OPF problem with input $(\boldsymbol{d}_0,\boldsymbol{d}_1,\cdots,\boldsymbol{d}_K)$. Therefore, we can prove that $(\boldsymbol{\zeta}_0^*,\boldsymbol{\zeta}_{\pi_1}^*,\cdots,\boldsymbol{\zeta}_{\pi_K}^*)$ is the optimal solution to the two-stage stochastic AC-OPF problem with input $(\boldsymbol{d}_0,\boldsymbol{d}_{\pi_1},\cdots,\boldsymbol{d}_{\pi_K})$. Thus, we can show that the input to first-stage solution mapping is partially permutation-invariant with respect to any permutation to the second-stage scenarios.

□

Appendix B
Proof of Theorem 3

Proof. (Sketch) For some $b_\phi\in\left[K(m+1),K^5m^2\right]$, we first define $\Psi(\boldsymbol{d}_1,\ldots,\boldsymbol{d}_K)=\sum_{i=1}^K\phi(\boldsymbol{d}_i)$, where ϕ is defined as: $\phi(\boldsymbol{d}_i)=\left[\psi_K\left(\boldsymbol{w}_1^T\boldsymbol{d}_i\right)^T,\ldots,\psi_K\left(\boldsymbol{w}_M^T\boldsymbol{d}_i\right)^T\right]$, for some $\boldsymbol{w}_1,\ldots,\boldsymbol{w}_M\in\mathbb{R}^m$, $M=b_\phi/K$, and $\psi_K(x)=\left[x,x^2,\ldots,x^K\right]$. We proceed by defining the image space of $\Psi(\boldsymbol{d}_1,\ldots,\boldsymbol{d}_K)$ as $\boldsymbol{Z}=\{\Psi(\boldsymbol{d}_1,\ldots,\boldsymbol{d}_K):\boldsymbol{d}_k\in\mathbb{R}^m,\forall 1\leq k\leq K\}$. According to [67], the mapping Ψ is injective with a continuous inverse. Therefore, we can identify a continuous mapping ρ with $\rho\left(\boldsymbol{d}_0,\boldsymbol{z}\right)=\mathcal{P}^*\left(\boldsymbol{d}_0,\Psi^{-1}(\boldsymbol{z})\right)$, given $\boldsymbol{z}\in\boldsymbol{Z}$. This ensures that $\rho\left(\boldsymbol{d}_0,\sum_{k=1}^K\phi(\boldsymbol{d}_k)\right)=\mathcal{P}^*\left(\boldsymbol{d}_0,\Psi^{-1}\circ\Psi(\boldsymbol{d}_1,\ldots,\boldsymbol{d}_K)\right)=\mathcal{P}^*(\boldsymbol{d}_0,\boldsymbol{d}_1,\ldots,\boldsymbol{d}_K)$. □

Appendix C
Proof of Theorem 5

Proof. We denote the input to the scenario-based two-stage stochastic AC-OPF problem as $\boldsymbol{l}=(\boldsymbol{d}_0,\boldsymbol{d}_1,\ldots,\boldsymbol{d}_K)\in\mathcal{D}$, where each $\boldsymbol{d}_k\in\mathbb{R}^m$ and $\mathcal{D}$ represents the load domain. We begin with proving that $\epsilon(b_\phi)$ is a lower bounded by ϵ_p. We first recall the definition of $\epsilon(b_\phi)$:

$$\epsilon(b_\phi)\overset{\triangle}{=}\inf_{\substack{\phi\in\mathcal{F}\\ \rho\in\mathcal{Q}}}\sup_{\boldsymbol{l}\in\mathcal{D}}\left\|\mathcal{P}^*(\boldsymbol{l})-\rho\left(\boldsymbol{d}_0,\sum_{k=1}^K\phi(\boldsymbol{d}_k)\right)\right\|_2^2, \tag{33}$$

where $\mathcal{P}^* : \mathbb{R}^{m(K+1)} \rightarrow \mathbb{R}^n$ is our target input to first-stage solution mapping in the scenario-based two-stage stochastic AC-OPF problem. $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^{b_\phi}$ and $\rho : \mathbb{R}^{m+b_\phi} \rightarrow \mathbb{R}^n$ are two continuous mappings. 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 mappings from $\mathbb{R}^{m+b_\phi}$ to $\mathbb{R}^n$. Given a PPNN model composed by neural networks ϕ_{nn} and ρ_{nn}, we recall the definition of the approximation error ϵ_p:

$$\epsilon_{\text{p}} \overset{\triangle}{=} \inf_{\substack{\phi\in\mathcal{H}\\ \rho\in\mathcal{V}}} \sup_{\boldsymbol{l}\in\mathcal{D}} \left\| \mathcal{P}^*(\boldsymbol{l}) - \rho\left(\boldsymbol{d}_0, \sum_{k=1}^{K} \phi(\boldsymbol{d}_k)\right)\right\|_2^2, \tag{34}$$

where $\mathcal{H}$ and $\mathcal{V}$ are the set of functions represented by ϕ_{nn} and ρ_{nn}, respectively.

As the input-output mapping of neural networks is continuous, we can show that $\mathcal{H}$ is a subset $\mathcal{F}$ and $\mathcal{V}$ is a subset of $\mathcal{Q}$. Thus, we can show that:

$$\begin{aligned}
\epsilon_{\text{p}} &= \inf_{\substack{\phi\in\mathcal{H}\\ \rho\in\mathcal{V}}} \sup_{\boldsymbol{l}\in\mathcal{D}} \left\| \mathcal{P}^*(\boldsymbol{l}) - \rho\left(\boldsymbol{d}_0, \sum_{k=1}^{K} \phi(\boldsymbol{d}_k)\right)\right\|_2^2 && (35)\\
&\geq \inf_{\substack{\phi\in\mathcal{F}\\ \rho\in\mathcal{Q}}} \sup_{\boldsymbol{l}\in\mathcal{D}} \left\| \mathcal{P}^*(\boldsymbol{l}) - \rho\left(\boldsymbol{d}_0, \sum_{k=1}^{K} \phi(\boldsymbol{d}_k)\right)\right\|_2^2 && (36)\\
&= \epsilon(b_\phi). && (37)
\end{aligned}$$

This shows that ϵ_p is lower bounded by $\epsilon(b_\phi)$. We then show the ϵ_p is upper bounded by $\epsilon(b_\phi) + K\,\text{Lip}(\rho^*)\epsilon_{\phi_{\text{nn}}} + \epsilon_{\rho_{\text{nn}}}$:

$$\begin{aligned}
\epsilon_{\text{p}} &= \inf_{\substack{\phi\in\mathcal{H}\\ \rho\in\mathcal{V}}} \sup_{\boldsymbol{l}\in\mathcal{D}} \left\| \mathcal{P}^*(\boldsymbol{l}) - \rho\left(\boldsymbol{d}_0, \sum_{i=1}^{K} \phi(\boldsymbol{d}_i)\right)\right\|_2^2 && (38)\\
&= \inf_{\substack{\phi\in\mathcal{H}\\ \rho\in\mathcal{V}}} \sup_{\boldsymbol{l}\in\mathcal{D}} \left\| \mathcal{P}^*(\boldsymbol{l}) - \rho^*\left(\boldsymbol{d}_0, \sum_{i=1}^{K} \phi^*(\boldsymbol{d}_i)\right)\right. \\
&\quad \left. + \rho^*\left(\boldsymbol{d}_0, \sum_{i=1}^{K} \phi^*(\boldsymbol{d}_i)\right) - \rho\left(\boldsymbol{d}_0, \sum_{i=1}^{K} \phi(\boldsymbol{d}_i)\right)\right\|_2^2 && (39)\\
&\leq \inf_{\substack{\phi\in\mathcal{H}\\ \rho\in\mathcal{V}}} \sup_{\boldsymbol{l}\in\mathcal{D}} \left\| \mathcal{P}^*(\boldsymbol{l}) - \rho^*\left(\boldsymbol{d}_0, \sum_{i=1}^{K} \phi^*(\boldsymbol{d}_i)\right)\right\|_2^2 + \\
&\quad \inf_{\substack{\phi\in\mathcal{H}\\ \rho\in\mathcal{V}}} \sup_{\boldsymbol{l}\in\mathcal{D}} \left\| \rho^*\left(\boldsymbol{d}_0, \sum_{i=1}^{K} \phi^*(\boldsymbol{d}_i)\right) - \rho\left(\boldsymbol{d}_0, \sum_{i=1}^{K} \phi(\boldsymbol{d}_i)\right)\right\|_2^2 && (40)\\
&\leq \epsilon(b_\phi) + \inf_{\phi\in\mathcal{H}} \sup_{\boldsymbol{l}\in\mathcal{D}} \left(\left\| \rho^*\left(\boldsymbol{d}_0, \sum_{i=1}^{K} \phi^*(\boldsymbol{d}_i)\right) - \rho^*\left(\boldsymbol{d}_0, \sum_{i=1}^{K} \phi(\boldsymbol{d}_i)\right)\right\|_2^2 \right. \\
&\quad \left. + \left\| \rho^*\left(\boldsymbol{d}_0, \sum_{i=1}^{K} \phi(\boldsymbol{d}_i)\right) - \rho\left(\boldsymbol{d}_0, \sum_{i=1}^{K} \phi(\boldsymbol{d}_i)\right)\right\|_2^2 \right) && (41)\\
&\leq \epsilon(b_\phi) + K\,\text{Lip}(\rho^*)\epsilon_{\phi_{\text{nn}}} + \epsilon_{\rho_{\text{nn}}}. && (42)
\end{aligned}$$

Therefore, we can prove that the approximation error of PPNN is upper bounded by $\epsilon(b_\phi) + K\,\text{Lip}(\rho^*)\epsilon_{\phi_{\text{nn}}} + \epsilon_{\rho_{\text{nn}}}$. $\square$

Appendix D
Chance-constrained AC-OPF

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

$$\begin{aligned}
\min\ & \sum_{i\in\mathcal{B}} c_{i,1}\cdot (p^g_{i0})^2 + c_{i,2}\cdot p^g_{i0} + c_{i,3} \\
& + \sum_{i\in\mathcal{B}} \mathbb{E}_{\boldsymbol{\xi}}\left[c_{i,1}\cdot (p^g_i(\boldsymbol{\xi}))^2 + c_{i,2}\cdot p^g_i(\boldsymbol{\xi}) + c_{i,3}\right] && (43)\\
\text{s.t. } & p^f_{ij0} = g_{ij}v_{i0}^2 - v_{i0}v_{j0}(g_{ij}\cos\theta_{ij0} + b_{ij}\sin\theta_{ij0}), (i,j)\in\mathcal{E}, && (44)\\
& q^f_{ij0} = -b_{ij}v_{i0}^2 - v_{i0}v_{j0}(g_{ij}\sin\theta_{ij0} - b_{ij}\cos\theta_{ij0}), (i,j)\in\mathcal{E}, && (45)\\
& p^g_{i0} - p^d_{i0} = \sum\nolimits_{(i,j)\in\mathcal{E}} p^f_{ij0},\ i\in\mathcal{B}, && (46)\\
& q^g_{i0} - q^d_{i0} = \sum\nolimits_{(i,j)\in\mathcal{E}} q^f_{ij0},\ i\in\mathcal{B}, && (47)\\
& \underline{p^g_i} \leq p^g_{i0} \leq \overline{p^g_i},\ i\in\mathcal{B}, && (48)\\
& \underline{q^g_i} \leq q^g_{i0} \leq \overline{q^g_i},\ i\in\mathcal{B}, && (49)\\
& \underline{v_i} \leq v_{i0} \leq \overline{v_i},\ i\in\mathcal{B}, && (50)\\
& \underline{\theta_{ij}} \leq \theta_{ij0} = \theta_{i0} - \theta_{j0} \leq \overline{\theta_{ij}},\ (i,j)\in\mathcal{E}, && (51)\\
& (p^f_{ij0})^2 + (q^f_{ij0})^2 \leq (\overline{s_{ij}})^2, (i,j)\in\mathcal{E}, && (52)\\
& p^f_{ij}(\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, && (53)\\
& q^f_{ij}(\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, && (54)\\
& p^g_i(\hat{\xi}) - p^d_i(\hat{\xi}) = \sum\nolimits_{(i,j)\in\mathcal{E}} p^f_{ij}(\hat{\xi}),\ i\in\mathcal{B},\ \hat{\xi}\in\Xi, && (55)\\
& q^g_i(\hat{\xi}) - q^d_i(\hat{\xi}) = \sum\nolimits_{(i,j)\in\mathcal{E}} q^f_{ij}(\hat{\xi}),\ i\in\mathcal{B},\ \hat{\xi}\in\Xi, && (56)\\
& \mathbb{P}_{\boldsymbol{\xi}}\begin{Bmatrix}
\underline{p^g_i} \leq p^g_i(\boldsymbol{\xi}) \leq \overline{p^g_i}, i\in\mathcal{B}, \\
\underline{q^g_i} \leq q^g_i(\boldsymbol{\xi}) \leq \overline{q^g_i}, i\in\mathcal{B}, \\
\underline{v_i} \leq v_i(\boldsymbol{\xi}) \leq \overline{v_i}, i\in\mathcal{B}, \\
\underline{\theta_{i,j}} \leq \theta_i(\boldsymbol{\xi}) - \theta_j(\boldsymbol{\xi}) \leq \overline{\theta_{ij}}, (i,j)\in\mathcal{E}, \\
(p^f_{i,j}(\boldsymbol{\xi}))^2 + (q_{i,j}(\boldsymbol{\xi})^f)^2 \leq (\overline{s_{i,j}})^2, (i,j)\in\mathcal{E}, \\
|p^g_i(0) - p^g_i(\boldsymbol{\xi})| \leq \Delta p_i, i\in\mathcal{B},
\end{Bmatrix} \geq \delta. && (57)\\
\text{var. } & p^g_{i0}, q^g_{i0}, \theta_{i0}, v_{i0}, p^g_i(\hat{\xi}), q^g_i(\hat{\xi}), \theta_i(\hat{\xi}), v_i(\hat{\xi}),\ i\in\mathcal{B}, \hat{\xi}\in\Xi. && (58)
\end{aligned}$$

Here, $\boldsymbol{\xi}$ denotes the random load in the second stage, with support set Ξ. $\hat{\xi}$ is a realization of $\boldsymbol{\xi}$. $p^d_i(\hat{\xi})$ and $q^d_i(\hat{\xi})$ are the active and reactive load at bus i under demand $\hat{\xi}$. $p^g_i(\hat{\xi})$ and $q^g_i(\hat{\xi})$ 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 $\hat{\xi}$. $p^f_{ij}(\hat{\xi})$ and $q^f_{ij}(\hat{\xi})$ are the active and reactive power flow at line (i,j) under realization $\hat{\xi}$.

Equations (44) and (45) define the active and reactive power flow in the first stage. The power balance in the first stage is ensured by equations (46) and (47). Constraints (48) and (49) 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 (50) and (51). Constraint (52) ensures the branch flow limit in the first stage. Equations (53) and (54) 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 equations (55) and (56). Constraint (57) 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.

Appendix E
Second-Stage Sub-Problem

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

$$
\begin{aligned}
\min\ & \sum\nolimits_{i\in\mathcal{B}} c_{i1}\cdot(p_i^g)^2 + c_{i2}\cdot p_i^g + c_{i3} && (59)\\
\text{s.t. } & p_{ij}^f = g_{ij}v_i^2 - v_iv_j(g_{ij}\cos\theta_{ij} + b_{ij}\sin\theta_{ij}), (i,j)\in\mathcal{E}, && (60)\\
& q_{ij}^f = -b_{ij}v_i^2 - v_iv_j(g_{ij}\sin\theta_{ij} - b_{ij}\cos\theta_{ij}), (i,j)\in\mathcal{E}, && (61)\\
& p_i^g - p_{ik}^d = \sum\nolimits_{(i,j)\in\mathcal{E}} p_{ij}^f, i\in\mathcal{B}, && (62)\\
& q_i^g - q_{ik}^d = \sum\nolimits_{(i,j)\in\mathcal{E}} q_{ij}^f,\ i\in\mathcal{B}, && (63)\\
& \underline{p_i^g} \le p_i^g \le \overline{p_i^g},\ i\in\mathcal{B}, && (64)\\
& \underline{q_i^g} \le q_i^g \le \overline{q_i^g},\ i\in\mathcal{B}, && (65)\\
& \underline{v_i} \le v_i \le \overline{v_i},\ i\in\mathcal{B}, && (66)\\
& \underline{\theta_{ij}} \le \theta_{ij} = \theta_i - \theta_j \le \overline{\theta_{ij}}, (i,j)\in\mathcal{E}, && (67)\\
& (p_{ij}^f)^2 + (q_{ij}^f)^2 \le (\overline{s_{ij}})^2, (i,j)\in\mathcal{E}, && (68)\\
& |p_i^g - p_{i0}^{g*}| \le \Delta p_i,\ i\in\mathcal{B}, && (69)\\
\text{var. } & p_i^g, q_i^g, \theta_i, v_i,\ i\in\mathcal{B}. && (70)
\end{aligned}
$$

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).

Equations (60) and (61) define the active and reactive power flow. The power balance is ensured by equations (62) and (63). Constraints (64) and (65) capture the active and reactive power generation limits. The voltage magnitude and phase angle limits are described in constraints (66) and (67). Constraint (68) ensures the branch flow limit. Constraint (69) ensures the ramping limits on active power generation. The objective is to minimize the second-stage generation cost.

Appendix F
Solution technique of MIPS for the two-stage stochastic AC-OPF problem

The primal-dual interior-point method employs the primal-dual interior-point method through the following systematic procedure: (i) Transforms the original problem into a sequence of unconstrained problems using barrier functions with control parameter t; (ii) Constructs the Lagrangian incorporating the barrier function; (iii) Derives and linearizes the KKT conditions; (iv) Applies Newton's method to solve the linearized KKT systems for primal-dual variable updates; (v) Incrementally increases t to enforce constraint satisfaction. This iterative process continues until KKT conditions are satisfied within specified tolerances, indicating convergence to a potential (local) minimum.

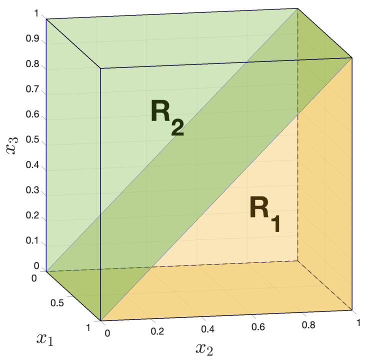

Fig. 5. An illustrative example for the partition of the input space. Consider a target mapping $\mathcal{P}(\boldsymbol{x}) = x_1 + \frac{1}{2}(x_2 + x_3)$, where the input domain is $[0,1]^3$. Here, mapping $\mathcal{P}$ is partially permutation-invariant to x_2 and x_3. Due to this property, we can partition the input domain into 2! regions: $R_1 = \{(x_1,x_2,x_3)\in[0,1]^3 \mid x_2 > x_3\}$ and $R_2 = \{(x_2,x_3)\in[0,1]^3 \mid x_2 < x_3\}$. For any point $\boldsymbol{x} = (x_1,x_2,x_3)$ in R_1, we can find it permuted variant $\boldsymbol{x}' = (x_1,x_3,x_2)\in R_2$ through permutation, and vice versa. By exploiting the partial permutation-invariance property of $\mathcal{P}$, we can focus the training of PPNN, denoted as $\hat{\mathcal{P}}$, solely on region R_2 (or equivalently R_1). For any $\boldsymbol{x}\in R_1$ outside the training region R_2, the partial permutation invariance of both $\mathcal{P}$ and $\hat{\mathcal{P}}$ implies $\mathcal{P}(\boldsymbol{x}) = \mathcal{P}(\boldsymbol{x}')$ and $\hat{\mathcal{P}}(\boldsymbol{x}) = \hat{\mathcal{P}}(\boldsymbol{x}')$. Therefore, the accuracy achieved on R_2 extends to R_1 without explicit training on that region.

Appendix G
Exploiting Partial Permutation-Invariance for Efficient Learning in the two-stage stochastic AC-OPF problem

Partial permutation-invariance is a natural and fundamental symmetric property in the two-stage stochastic AC-OPF problem, which means that the optimal first-stage decision does not change if we permute the order of second-stage scenarios. For an input with K second-stage scenarios, there are $K!$ possible permutations, but all of them map to the same first-stage solution. Based on this property, we can partition the input space into $K!$ equivalent regions, where for any two regions, the input in one region can be obtained through permutation by that of the other region. See Figure 5 for an example.

By designing partially permutation-invariant neural networks (PPNNs) to capture symmetry, we can simplify the learning task by training on one of the $K!$ equivalent regions. For an input $\boldsymbol{l}$ outside the trained region, its permuted variant $\boldsymbol{l}'$ within the trained region can be found. Due to the partial permutation invariance of both the target mapping and PPNN, $\boldsymbol{l}$ and $\boldsymbol{l}'$ share the same target solution, PPNN prediction, and accuracy. Thus, achieving a high accuracy in the trained region ensures the same accuracy across all regions, allowing us to learn the target mapping in just one of the $K!$ equivalent regions instead of the entire input space, greatly simplifying the task.

In contrast, vanilla designs like fully-connected neural networks (FCNNs) do not exploit this symmetry. Even if an FCNN achieves high accuracy for inputs in one of the $K!$ regions, it does not guarantee a high accuracy for input outside it, due to the lack of partial permutation-invariance. As a result, FCNNs must be trained over the entire input space, which is significantly larger than that of PPNN. This lack of symmetry exploitation makes FCNNs less efficient than PPNNs.

In summary, by exploiting the partial permutation-invariance property, we can reduce the effective size of the input space and simplify the learning task, leading to more efficient and accurate predictions compared to methods that do not capture this property.