Solving Non-linear Optimization Problem in Engineering by Model-Informed Generative Adversarial Network (MI-GAN)

Yuxuan Li
School of Industrial Engineering and Management
Oklahoma State University
Stillwater, Oklahoma, USA
yuxuan.li@okstate.edu

Chaoyue Zhao
Department of Industrial and Systems Engineering
University of Washington
Seattle, Washington, USA
cyzhao@uw.edu

Chenang Liu*
School of Industrial Engineering and Management
Oklahoma State University
Stillwater, Oklahoma, USA
chenang.liu@okstate.edu

2022 IEEE International Conference on Data Mining Workshops (ICDMW) | 979-8-3503-4609-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDMW58026.2022.00035

***Abstract*— Optimization models have been widely used in many engineering systems to solve the problems related to system operation and management. For instance, in power systems, the optimal power flow (OPF) problem, which is a critical component of power system operations, can be formulated using optimization models. Specifically, the alternating current OPF (AC-OPF) problems are challenging since some of the constraints are non-linear and non-convex. Moreover, due to the high variability that the power system may have, the coefficients of the optimization model may change, increasing the difficulty of solving the OPF problem. Although the conventional optimization tools and deep learning approaches have been investigated, the feasibility and optimality of the solutions may still be unsatisfactory. Hence, in this paper, based on the recently developed model-informed generative adversarial network (MI-GAN) framework, a tailored version for solving the non-linear AC-OPF problem under uncertainties is proposed. The contributions of this work can be summarized into two main aspects: (1) To ensure the feasibility and improve the optimality of the generated solutions, two important layers, namely, the feasibility filter layer and optimality-filter layer, are considered and designed; and (2) An efficient model-informed selector is designed and integrated to the GAN architecture, by incorporating these two new layers to inform the generator. Experiments on the IEEE test systems demonstrate the efficacy and potential of the proposed method for solving non-linear AC-OPF problems.**

***Keywords*-- generative adversarial network (GAN), model-informed generation, optimal power flow (OPF), non-linear optimization**

I. INTRODUCTION

Many problems in system operations and management can be formulated and handled by optimization models in engineering practice. For example, in the operations of power systems, one of its essential components is to estimate the optimal power flow (OPF), which aims to minimize the total generation cost while satisfying the operational requirements and system constraints [1, 2]. A common approach to deal with OPF problem is to formulate it as an optimization model and search for the optimal solution [3, 4]. Specifically, the alternating current OPF (AC-OPF) formulation is widely applied in real-world power system operations. However, it is notoriously challenging to be solved due to its nonlinear and nonconvex nature. Also, the AC-OPF models are usually not fixed, i.e., model parameters may change dynamically with uncertainties in the power system [5]. Though some traditional optimization approaches, such as stochastic and robust methods [5], could address the OPF problem effectively, the problems are solved individually in each stage, which is time-consuming and not suitable for online applications.

In recent decades, an increasing number of artificial intelligent-based techniques have been developed to solve the optimization problems [6-9]. The majority of them are based on neural networks, such as convolutional neural network [6], deep neural network [7], and graph neural network [8]. The neural networks are trained to predict the sets of independent operational variables. In this way, the number of variables could be considerably lowered, reducing the complexity of neural network structures and speeding up the computation. However, massive simulations need to be obtained in advance for model training, which may result in sample inefficiency.

To close the gaps mentioned above, a natural idea is to search for optimal solution based on the historical solutions instead of solving the entire optimization model at each stage. In addition, an efficient way to train the neural networks ahead of time is also essential to be considered. Following in this vein, Li *et al.* [9] developed a model-informed generative network to handle the linearly modeled direct current OPF (DC-OPF) problem. This framework is based on generative adversarial network (GAN), which was first proposed by Goodfellow *et al.* [10]. GAN has been recently used in a large variety of applications, including image generation [11], missing data imputation [12], manufacturing process monitoring [13], biology [14], and so on. The GAN is composed of two networks, generator G and discriminator D, which is based on a minimax game for $V(D,G)$ shown in (1).

*Corresponding author: Chenang Liu

2375-9259/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDMW58026.2022.00035

$$\min_{G}\max_{D} V(D,G_m) = \mathbb{E}_{\mathbf{x}\sim P_{\mathbf{dddd}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim P_{\mathbf{Z}}(\mathbf{z})}[\log(1-D(G(\mathbf{z})))] \quad (1)$$

where G generates artificial samples $G(\mathbf{z})$ and D distinguishes between $G(\mathbf{z})$ and actual samples $\mathbf{x}$. G and D compete with each other until D is unable to differentiate actual data and artificial data. In such cases, the distribution of artificial data should be highly similar to that of actual data [10].

In the optimization fields, model-informed GAN (MI-GAN) [9] is developed to learn the distribution of data around optimal solutions. In MI-GAN, G generates the solutions and D distinguishes the generated solutions and historical solutions, where the historical solutions are approximately centered on the optimal solution. More importantly, a model-informed selector is proposed to ensure that the solutions generated by G are able to satisfy the model constraints.

In this paper, using the MI-GAN architecture, we further tailored a more advanced MI-GAN framework to solve the non-linear optimization problems under uncertainties, e.g., AC-OPF. Its main contribution consists of two aspects:

i) Two critical model-informed layers are designed, feasibility filter layer and optimality filter layer, to guarantee the feasibility and optimize the generated solutions.
ii) An efficient model-informed selector is built based on these two model-informed layers in the generator of GAN-based architecture.

Furthermore, the performance of the proposed MI-GAN based approach is also demonstrated in the AC-OPF problem. The remainder of the work is organized as follows: In Section II, the revised MI-GAN is proposed to solve non-linear optimization problem. Numerical case study results are presented in Section V to demonstrate the effectiveness of the proposed method. Finally, conclusions and future work are discussed in Section VI.

II. Research Methodology

A. Model-informed GAN (MI-GAN)

As shown in Fig. 1, MI-GAN is incorporated into the optimization process when training the neural networks. Similar to GAN, there are two key components in MI-GAN, i.e., the model-informed (MI) generator G_m and the discriminator D. G_m is to generate solution sets, i.e., $\mathbf{X}_u$, whereas D is to distinguish the generated solution sets, i.e., $\mathbf{X}_u$, and historical solution sets, i.e., $\mathbf{X}_h$. When the model converges, the distribution of the generated solutions, i.e., P_{G_m}, should be the same as the distribution of the historical solutions, i.e., $P_{\mathbf{dddd}}$.

The objective function is denoted by $f(\cdot)$. When $\mathbf{X}_h$ is close to the optimal solution, i.e., $\mathbf{x}^*$, the objective function values calculated by $\mathbf{X}_h$, i.e., $f(\mathbf{X}_h)$, should also be close to the optimal objective function value, i.e., $f(\mathbf{x}^*)$. Hence, since $P_{G_m} = P_{\mathbf{dddd}}$ is achieved when the model converges, $\mathbf{X}_u$ should be similar to $\mathbf{X}_h$. That is, $f(\mathbf{X}_u) \approx f(\mathbf{X}_h)$. In this way, the generated solutions will be similar or close to the optimal solutions. Notably, $\mathbf{X}_h$ could be selected to feed D from the optimal or near-optimal solutions of the previous or solved optimization models.

Besides, in the model-informed (MI) selector M, $\mathbf{X}$ is also updated by identifying the feasibility and optimality of the generated solutions. For comparison, a saved solution set, i.e., $\mathbf{X}_s$, with the same size as $\mathbf{X}$, is used for comparison. In each iteration, $\mathbf{X}_s$ will be replaced by $\mathbf{X}_u$ to save the generated solutions.

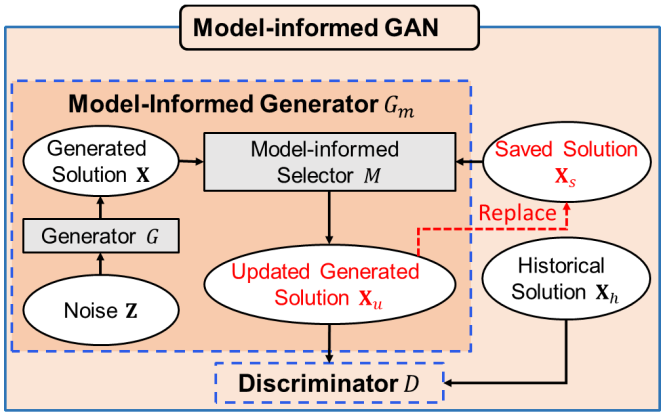

Fig. 1. A demonstration of the proposed MI-GAN based framework.

B. Model-informed Selector

As stated in Sec. II.A, the MI selector M updates the solutions based on the comparison between $\mathbf{X}$ and $\mathbf{X}_s$. The comparison considers two distinct factors: feasibility and optimality. Hence, two different model-informed layers, namely, feasibility filter layer and optimality filter layer, are proposed to construct M as illustrated in Fig. 2.

In Fig. 2, there are two steps in M to update $\mathbf{X}$. First, the feasibility filter layer is used to ensure that the constraints for both $\mathbf{X}$ and $\mathbf{X}_s$ are satisfied. The solutions in $\mathbf{X}$ may be replaced by the solutions in $\mathbf{X}_s$ depending on the different feasibility conditions. In this way, the new generated solution set, i.e., $\mathbf{X}_n$, can be obtained. Afterwards, the optimality filter layer is applied to compute the objective function values for both $\mathbf{X}_n$ and $\mathbf{X}_s$. The solutions in $\mathbf{X}_n$ may also be replaced by the solutions in $\mathbf{X}_s$ depending on different optimality conditions. The updated generated solution set, i.e., $\mathbf{X}_n$, could be then obtained.

1) Feasibility filter layer

Fig. 3(a) provides an illustration of the proposed feasibility filter layer. Initially, $\mathbf{X}$ and $\mathbf{X}_s$ are combined as pairs and then sent to this layer. That is, the pair involves one solution from $\mathbf{X}$ and another solution from $\mathbf{X}_s$. Therefore, for the i-th pair, i.e., $(\mathbf{X}^{(i)}, \mathbf{X}_s^{(i)})$, $i = 1,2,\dots,n$, one solution will be selected from this pair and then passed to the newly generated solution set $\mathbf{X}_n$.

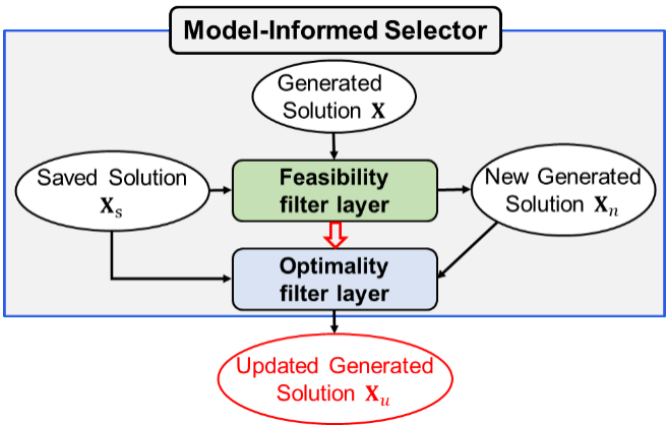

Fig. 2. An overview of the model-informed selector.

In particular, based on the feasibility conditions of both solutions, $(\mathbf{X}^{(i)}, \mathbf{X}_s^{(i)})$ may have four cases: i) feasible $\mathbf{X}^{(i)}$ and infeasible $\mathbf{X}_s^{(i)}$; ii) feasible $\mathbf{X}^{(i)}$ and feasible $\mathbf{X}_s^{(i)}$; iii) infeasible $\mathbf{X}^{(i)}$ and infeasible $\mathbf{X}_s^{(i)}$; as well as iv) infeasible $\mathbf{X}^{(i)}$ and feasible $\mathbf{X}_s^{(i)}$. To save the feasible solutions, for case i) and iv), the feasible solution will be passed to $\mathbf{X}_n$. However, for case ii) and iii), the feasibility of both solutions is the same. In such cases, the passed solution should be used to demonstrate the performance of G_m in current iteration. Hence, the generated solution is passed to $\mathbf{X}_n$ in case ii) and iii). Notably, the feasibility of each pair is also recorded to show which feasibility case it belongs to, which is denoted as S_i ($S_i = 1,2,3,4$) in one label sequence S. Then S will also be routed to the optimality filter layer.

Based on the above-mentioned description for feasibility filter layer, the corresponding algorithm is shown in the Algorithm 1. For the i-th pair, only when $\mathbf{X}^{(i)}$ is infeasible and $\mathbf{X}_s^{(i)}$ is feasible, $\mathbf{X}^{(i)}$ is passed to $\mathbf{X}_n$ as $\mathbf{X}_n^{(i)}$. Otherwise, $\mathbf{X}^{(i)}$ is assigned as $\mathbf{X}_n^{(i)}$. In this way, all the solutions in $\mathbf{X}_n$ are essentially synthesized from either current iteration or the previous iterations.

Algorithm 1: Feasibility filter layer algorithm

Input: $\mathbf{X}$ $\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(n)}\}$, $\mathbf{X}_s$ $\{\mathbf{X}_s^{(1)}, \mathbf{X}_s^{(2)}, \ldots, \mathbf{X}_s^{(n)}\}$
Step 1: Define a new solution set as $\mathbf{X}_n$ and a zero sequence S
For $i = 1$ **to** n **do**
  Step 2: Calculate the constraints based on $\mathbf{X}^{(i)}$ and $\mathbf{X}_s^{(i)}$
  If $\mathbf{X}^{(i)}$ is feasible and $\mathbf{X}_s^{(i)}$ is infeasible:
    Step 3: Assign $\mathbf{X}^{(i)}$ to $\mathbf{X}_n^{(i)}$ and set $S_i = 1$
  Else if $\mathbf{X}^{(i)}$ and $\mathbf{X}_s^{(i)}$ are both feasible:
    Step 4: Assign $\mathbf{X}^{(i)}$ to $\mathbf{X}_n^{(i)}$ and set $S_i = 2$
  Else if $\mathbf{X}^{(i)}$ and $\mathbf{X}_s^{(i)}$ are both infeasible:
    Step 5: Assign $\mathbf{X}^{(i)}$ to $\mathbf{X}_n^{(i)}$ and set $S_i = 3$
  Else: Assign $\mathbf{X}_s^{(i)}$ to $\mathbf{X}_n^{(i)}$ and set $S_i = 4$
Output $\mathbf{X}_n$, S

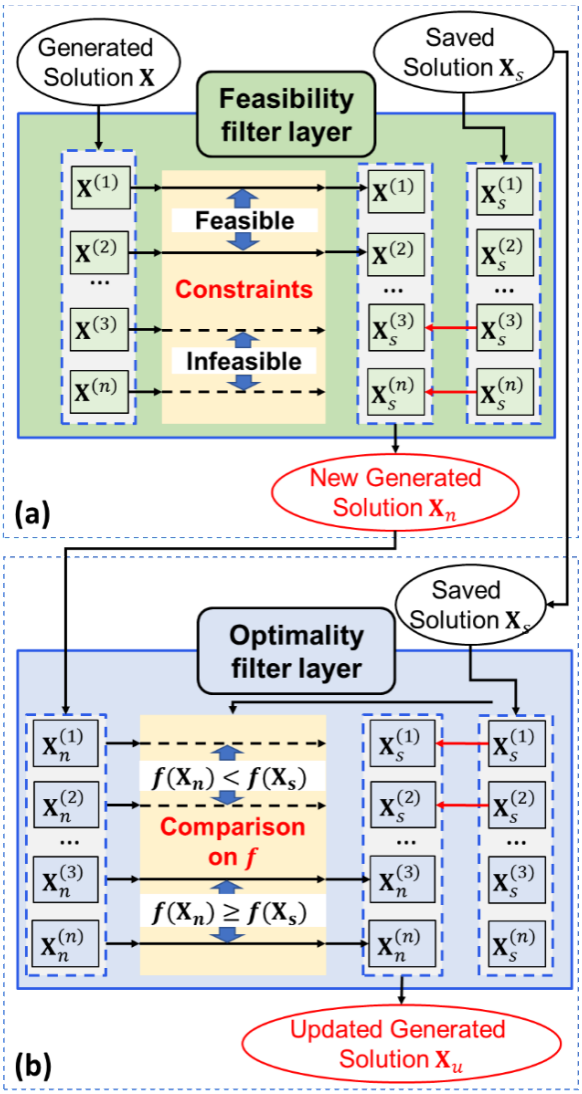

Fig. 3. The framework of the feasibility filter layer (a) and optimality filter layer (b) in the model-informed selector.

2) Optimality filter layer

The optimality filter layer compares the objective function values calculated by $\mathbf{X}_n$ and $\mathbf{X}_s$, i.e., $f(\mathbf{X}_n)$ and $f(\mathbf{X}_s)$. Fig. 3(b) depicts the optimality filter layer in M. Similar as the input of feasibility filter layer, $\mathbf{X}_n$ and $\mathbf{X}_s$ are also sent as pairs to the optimality filter layer. One solution in the pair is from $\mathbf{X}_n$ while another is from $\mathbf{X}_s$. The output of optimality filter layer, i.e., $\mathbf{X}_u$, will also select one solution from each pair.

The i-th pair of $\mathbf{X}_n$ and $\mathbf{X}_s$, i.e., $(\mathbf{X}_s^{1(i)}, \mathbf{X}_s^{(i)})$, may have two cases, as shown in (2).

$$\text{i) } f(\mathbf{X}_n^{(i)}) \geq f(\mathbf{X}_s^{(i)}); \text{ ii) } f(\mathbf{X}_n^{(i)}) < f(\mathbf{X}_s^{(i)}). \quad (2)$$

In such cases, a natural idea is to pass the solution with a smaller objective function value to $\mathbf{X}_u$. However, since the feasibility is more important than optimality, the feasibility of $\mathbf{X}_n$ and $\mathbf{X}_s$, i.e., the label S, should also be considered in the

optimality filter layer. Hence, case i) and ii) are discussed separately according to S:

(a) For the case i), if $\mathbf{X}_s^{(i)}$ is feasible, i.e., S_i is 1 or 2, $\mathbf{X}_s^{(i)}$ will be assigned as $\mathbf{X}_u^{(i)}$; Otherwise, $\mathbf{X}_n^{(i)}$ will be sent to $\mathbf{X}_s^2$ as $\mathbf{X}_s^{2(i)}$.

(b) For the case ii), only when $\mathbf{X}_s^{1(i)}$ is feasible and $\mathbf{X}_s^{(i)}$ is infeasible, i.e., S_i is 1, $\mathbf{X}_s^{(i)}$ will be assigned as $\mathbf{X}_u^{(i)}$, Otherwise, $\mathbf{X}_u^{(i)}$ will be sent to $\mathbf{X}_u$ as $\mathbf{X}_u^{(i)}$.

Based on the above two cases, the algorithm for optimality filter layer is demonstrated in Algorithm 2 below. The objective function values in both $\mathbf{X}_n$ and $\mathbf{X}_s$ are computed. Two cases are considered separately with the help of calculated objective function values and S. In this way, the feasible solutions with smaller objective function values could be passed to $\mathbf{X}_u$. Hence, based on the feasibility filter layer and optimality filter layer, the overall procedure of model-informed selector can be then established. The formulation of the proposed model is discussed in Sec. II-C.

Algorithm 2: Optimality filter layer algorithm

Input: $\mathbf{X}_n\{\mathbf{X}_n^{(1)}, \mathbf{X}_n^{(2)}, \dots, \mathbf{X}_n^{(n)}\}$, $\mathbf{X}_s\{\mathbf{X}_s^{(1)}, \mathbf{X}_s^{(2)}, \dots, \mathbf{X}_s^{(n)}\}$, S
Step 1: Define a new saved solution set as $\mathbf{X}_u$
For $i = 1$ **to** n **do**
  Step 2: Calculate $f(\mathbf{X}_n^{(i)})$ and $f(\mathbf{X}_s^{(i)})$
  If $f(\mathbf{X}_n^{(i)}) < f(\mathbf{X}_s^{(i)})$:
    If $S_i = 1$: Assign $\mathbf{X}_s^{(i)}$ to $\mathbf{X}_u^{(i)}$; Else: Assign $\mathbf{X}_n^{(i)}$ to $\mathbf{X}_u^{(i)}$
  Else if $S_i > 2$: Assign $\mathbf{X}_n^{(i)}$ to $\mathbf{X}_u^{(i)}$; Else: Assign $\mathbf{X}_s^{(i)}$ to $\mathbf{X}_u^{(i)}$
Output $\mathbf{X}_u$

C. Formulation of MI-GAN for Non-linear Optimization

Based on the description in Sec. II-B, the MI selector could be demonstrated as in (3). The generated solution set $\mathbf{X}$ is updated as $\mathbf{X}_u$ based on the saved solution set $\mathbf{X}_s$.

$$\mathbf{X}_u = M(\mathbf{X}|\mathbf{X}_s) \tag{3}$$

In this way, the model-informed generator, i.e., G_m, could be formulated in (4). The generator first generates solutions based on the input noise $\mathbf{Z}$. Afterwards, the generated solutions will be sent to the MI selector and output as $\mathbf{X}_u$. Then $\mathbf{X}_u$ will be sent to D with actual solution set $\mathbf{X}_h$ for distinguishing.

$$G_m(\mathbf{Z}) = M(G(\mathbf{Z})|\mathbf{X}_s) \tag{4}$$

Based on G_m and D, as shown in (5), the minimax game for the MI-GAN, i.e., $V(D, G_m)$, can be formulated in (5),

$$\min_{G_m}\max_{D} V(D, G_m) = \mathbb{E}_{\mathbf{x}_h \sim P_{\mathbf{dddd}}}[\log D(\mathbf{x}_h)] + \mathbb{E}_{\mathbf{Z}\sim P_{G_m}}[\log(1 - D(G_m(\mathbf{Z})))] \tag{5}$$

Accordingly, the loss of G_m, i.e., L_{G_m}, and the loss of D, L_D, could be expressed in (6).

$$\begin{aligned} L_{G_m} &= -\mathbb{E}_{\mathbf{z}\sim P_{G_m}}(D(G_m(\mathbf{Z}))) \\ L_D &= \mathbb{E}_{\mathbf{z}\sim P_{G_m}}(D(G_m(\mathbf{Z}))) - \mathbb{E}_{\mathbf{x}_h \sim P_{\mathbf{dddd}}}(D(\mathbf{x}_h)) \end{aligned} \tag{6}$$

Based on the established MI-selector, the algorithm for MI-GAN is shown in Algorithm 3. $\mathbf{X}_h$ and an initialized pre-defined $\mathbf{X}_s$ are sent to the MI-GAN. Then in first iteration, the pre-defined solutions in $\mathbf{X}_s$ could be replaced by the generated solutions $\mathbf{X}$ according to the principle described in Sec. II-A. Hence, in each iteration, the solutions in $\mathbf{X}_s$ are all generated either from the current iteration or from the previous iterations. Then, when L_D and L_{G_m} converge, $\mathbf{X}_u$ should be similar to $\mathbf{X}_h$. Under such circumstances, $f(\mathbf{X}_u)$ will also be similar to $f(\mathbf{X}_h)$, which will be close to the optimal objective function value, i.e., $f(\mathbf{x}^*)$.

Algorithm 3: MI-GAN algorithm to solve non-linear optimization

Input: $\mathbf{X}_h\{\mathbf{X}_h^{(1)}, \mathbf{X}_h^{(2)}, \dots, \mathbf{X}_h^{(m)}\}$, $\mathbf{X}_s\{\mathbf{X}_s^{(1)}, \mathbf{X}_s^{(2)}, \dots, \mathbf{X}_s^{(n)}\}$, $\mathbf{c}$, $\mathbf{A}$, $\mathbf{b}$, f, parameter η, n
Repeat
  Step 1: Randomly generate noise $\mathbf{Z}$
  Step 2: Generate n fake solutions by generator as $\mathbf{X}\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \dots, \mathbf{X}^{(n)}\}$
  Step 3: Obtain $\mathbf{X}_u$ by inputting $\mathbf{X}_h$, f to **algorithm 1** and **algorithm 2**
  Step 4: Send $\mathbf{X}_h$ and $\mathbf{X}_u$ into discriminator D to get output label results $D(\mathbf{X})$ and $D(\mathbf{Z})$, respectively
  Step 5: Optimize the model parameters based on the output of discriminator
  Step 6: Assign $\mathbf{X}_u$ to $\mathbf{X}_s$
Until both Loss_{G_m}, $-D(\mathbf{X})$, and Loss_D, $D(\mathbf{X}) - D(\mathbf{Z})$, converge
  Output G_m, D

Specifically, to ensure that the solutions in the pre-defined $\mathbf{X}_s$ are replaced in the first iteration, each solution is set as a vector with extremely large values. In this way, in the feasibility filter layer, since the solutions in the pre-defined $\mathbf{X}_s$ are not feasible, they could be replaced by the generated solutions in $\mathbf{X}$. Hence, the pre-defined $\mathbf{X}_s$ will not interfere the training process of MI-GAN.

It is worth noting that, the proposed MI-GAN based framework has the same convergence property like MI-GAN. That is, when $P_{\mathbf{dddd}} = P_{G_m}$ [9], the model will converge. If $\mathbf{x}^*$ locates in the center of $P_{\mathbf{dddd}}$, then after the model converges, the solutions from G_m, i.e., $\mathbf{X}_u$, should also be around $\mathbf{x}^*$. Following that, the solution with the smallest objective function values among $\mathbf{X}_u$ is selected as the output solution of MI-GAN model, i.e., $\bar{\mathbf{x}}$. To achieve that, the historical solutions of the optimization model could be used as $\mathbf{X}_h$.

If the historical solutions are not available, solutions can be sampled from the feasible region. Then the sampled solutions could also be considered as historical solutions. When the optimization model is large in scale, such sampling may take a long time. In such cases, the constraints could be partially relaxed in order to reduce the sampling time.

In addition, when the sampled solutions are applied as $\mathbf{X}_h$, $\mathbf{x}^*$ may not be located in the center of $P_{\mathbf{dddd}}$. Then $\mathbf{X}_u$ may also not be around $\mathbf{x}^*$. Under such conditions, the recursive iteration algorithm, similar as the process in [9], could be developed to make $\mathbf{X}_u$ approach to $\mathbf{x}^*$. That is, the historical solutions will be replaced by the generated solutions gradually and the model will be trained based on the updated actual solutions.

III. Case Studies

A. Data Introduction and Experimental Setup

In this paper, an IEEE test system for alternating current optimal power flow (AC-OPF) problem is used to validate the effectiveness of the proposed method. As described in Sec. I, AC-OPF problem is a very popular non-linear optimization problem in power system operations and management. It minimizes the total generation cost while satisfying system constraints and operational requirements. In this way, economic power system operation could be achieved in day-ahead and real-time markets [1].

The AC-OPF problem involves four groups of variables: the actual power generation, $\boldsymbol{P}_g$, the reactive power generation, $\boldsymbol{Q}_g$, the power voltage magnitudes, $\boldsymbol{V}$, and the power voltage phase angles, $\boldsymbol{\theta}$. Denote N and N_G as the buses and power generators, respectively. Besides, n_b and n_g are used to represent the number of buses and power generators. Then in $\boldsymbol{P}_g$=$\{P_{g1}, P_{g2},\ldots, P_{gn_g}\}$, P_{gg} is the actual power generation of generator i, and in $\boldsymbol{Q}_g$=$\{Q_{g1}, Q_{g2},\ldots, Q_{gn_g}\}$, Q_{gg} is the reactive power generation of generator i. In addition, in $\boldsymbol{\theta}$=$\{\theta_1, \theta_2,\ldots, \theta_{n_b}\}$, θ_g is the voltage phase angle for bus i, and in $\boldsymbol{V}$=$\{V_1, V_2,\ldots, V_{n_b}\}$, V_g is the voltage magnitude for bus i. Similarly, denote that the actual power load and reactive power load as $\boldsymbol{P}_d$=$\{P_{g1}, P_{g2},\ldots, P_{gn_b}\}$ and $\boldsymbol{Q}_d = \{Q_{g1}, Q_{g2}, \ldots, Q_{gn_b}\}$, respectively. Furthermore, the transmission line is represented by $\mathcal{E}$. The number of lines in $\mathcal{E}$ is denoted as n_{lgnl}. Based on the transmission lines between different buses, the flow limit matrix following one matrix $\mathbf{S}$, whose size is $N \times N$. According to the notations, the formulation of the applied AC-OPF problem in this study is shown in (7)-(14).

$$\min_{\boldsymbol{P}_g,\boldsymbol{Q}_g,\boldsymbol{V},\boldsymbol{\theta}} C(\boldsymbol{P}_g) \tag{7}$$

$$\text{s.t. } P_{gg} - P_{dg} = \sum_{j\in N} V_g V_j \left(g_{gj}\cos\theta_{gj} + b_{gj}\sin\theta_{gj}\right), i \in N \tag{8}$$

$$Q_{gg} - Q_{dg} = \sum_{j\in N} V_g V_j \left(g_{gj}\sin\theta_{gj} - b_{gj}\cos\theta_{gj}\right), i \in N \tag{9}$$

$$P_g^{\min} \le P_{gg} \le P_{gg}^{\max}, i \in N_G \tag{10}$$

$$Q_{gg}^{\min} \le Q_{gg} \le Q_{gg}^{\max}, i \in N_G \tag{11}$$

$$V_g^{\min} \le V_g \le V_g^{\max}, i \in N \tag{12}$$

$$P_{gj}^2 + Q_{gj}^2 \le \left(S_{gj}^{\max}\right)^2, (i,j) \in \mathcal{E} \tag{13}$$

$$\theta_{gj}^{\min} \le \theta_{gj} \le \theta_{gj}^{\max}, (i,j) \in \mathcal{E} \tag{14}$$

Equation (7) minimizes the power generation cost where the cost function is denoted as C. Following that, equation (8)-(9) demonstrates the Kirchhoff's circuit laws for both actual power and reactive power, respectively, where g and b are coefficients. In addition, equations (10)-(11) represent the limits for active power generation, reactive power generation and voltage magnitudes, separately. Equation (12) shows the flow limit on each transmission line and (13) demonstrates the voltage phase angles limit where $\theta_{gj} = \theta_g - \theta_j$.

In this study, an IEEE case9 test system is demonstrated for the AC-OPF problem. The system setups in this work are similar to those in Venzke *et al.* [15]. There are 9 buses and 3 power generators in case9, i.e., $n_b = 9$, and $n_g = 3$. In addition, there are 9 transmission lines, i.e., $n_{lgnl} = 9$. In addition, the cost function in this case is also linear.

As shown in (8)-(14), equation (8), (9) and (13) are non-linear constraints while the others are linear constraints. Hence, since the non-linear constraints are hard to handle, without loss of generality, the required variables are directly calculated rather than generated or sampled. That is, only $\boldsymbol{V}$ and $\boldsymbol{\theta}$ are synthesized for both sampling process for actual solutions and generation process for generated solutions. Then using $\boldsymbol{V}$ and $\boldsymbol{\theta}$, $\boldsymbol{P}_g$ and $\boldsymbol{Q}_g$ are calculated by (8) and (9). Following that, all the variables are sent to (10)-(14) for feasibility testing. This makes the optimization problem easier to learn. Specifically, in the sampling process for the actual solutions, 3,000 feasible pairs of $\boldsymbol{V}$ and $\boldsymbol{\theta}$ are sampled under the uniform distribution according to the range in (12) and (14).

Notably, the proposed MI-GAN is compatible with most of the popular GAN architectures. In this study, since the popular Wasserstein GAN (WGAN) [11] could make the training process more stable, MI-GAN is built by incorporating WGAN. Besides, in both D and G_m of MI-GAN, the multilayer perceptron (MLP) networks are applied. D consists of three fully connected layers whereas G_m consists of five fully connected layers. In the last layer of G_m, to customize the generated pairs of $\boldsymbol{V}$ and $\boldsymbol{\theta}$, they are mapped to the range following (12) and (14) after utilizing sigmoid as the activation function. For all the other layers in both D and G_m, Leaky_ReLU is incorporated as the activation function. The number of iterations is set as 3,000. The batching size is set as 50. That is, 50 feasible pairs of $\boldsymbol{V}$ and $\boldsymbol{\theta}$ are generated in the generator. The proposed method is implemented using the TensorFlow 1.13 [16]. The experiments were conducted on a regular computer with i7-9750H CPU @2.60 GHz, and 16 GB RAM.

B. Results and Discussions

In this study, to validate the effectiveness of the proposed method comprehensively, two different scenarios are considered for different perspectives. Sec. III-B-1 applied the fixed AC-OPF problems to show the performance of the proposed method. In addition, Sec. III-B-2 applied the AC-OPF problem with uncertainties to test the robustness of the proposed method.

1) MI-GAN performance under fixed AC-OPF problem

In this study, the output solution of the proposed method could be obtained from the first recursive iteration. Hence, unlike [9], the recursive iteration algorithm is not applied in this study. In addition, a convergence plot of the proposed method is shown in Fig. 4. That is, as the mode progresses from 0 to 2,000 iterations, the feasible solutions in the generated solution set will be recorded in each iteration. The average objective function values of generated feasible solutions are then calculated for plotting. Fig. 4. clearly shows that the objective function value gradually decreases and converges as the iteration progresses.

Specifically, according to Fig. 4, the update process of objective function values could be divided in two phases. In the first phase, which is before approximately 250 iterations, the feasibility filter layer plays a critical role. This is due to the MI-GAN model's inability to generate the desired solutions accurately during this phase. Hence, the model is primarily concerned with the feasibility filtering in this phase. Under these conditions, the number of feasible solutions in the generated solution set is increasing, but their quality is not guaranteed. Hence, the average objective function values may exhibit some fluctuation patterns during the first phase.

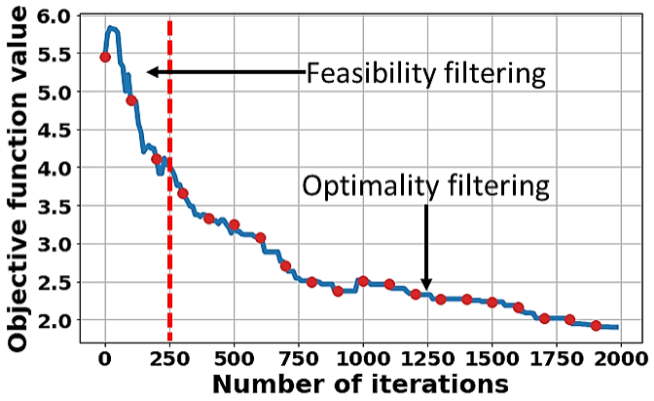

Fig. 4. The objective function value convergence for the proposed method.

In the second phase, which occurs after approximately 250 iterations, most of the solutions in the generated solution set are already feasible. That is, the MI-GAN model has already been well-trained to generate feasible solutions. Hence, the model is primarily concerned with the optimality filtering in this phase. Higher-cost objective values are replaced by the lower-cost objective values. In this way, the average objective values will decrease and converge steadily.

Based on the first phase and second phase in Fig. 4, the effectiveness of feasibility filter layer and optimality filter layer are validated. In addition, the generated solutions are approaching to the optimal solution. Thereby, it demonstrates that the proposed method could converge and gradually find the solutions around the optimal solution to solve this problem.

Besides, the experimental running time of MI-GAN were also recorded to demonstrate the efficiency of the proposed method. Specifically, the running time of the proposed model in each case is about 7.91 seconds, and the standard deviation of the running time is about 0.19. Hence, since the running time is relatively lower and the standard deviations are small, it has the potential to be applied in the online power systems.

2) MI-GAN performance under AC-OPF problem with uncertainties

In this study, to demonstrate the effectiveness of the proposed method comprehensively, the scenarios when the AC-OPF problems have uncertainties are demonstrated. That is, the experiments to add the fluctuations to the net loads, including $\boldsymbol{P}_d$ and $\boldsymbol{Q}_d$, are conducted. In the experiments, the proposed method is trained given the original $\boldsymbol{P}_d$ and $\boldsymbol{Q}_d$. After the proposed method is well-trained, $\boldsymbol{P}_d$ and $\boldsymbol{Q}_d$ are changed to $\rho\boldsymbol{P}_d$ and $\rho\boldsymbol{Q}_d$, respectively. Then according to $\rho\boldsymbol{P}_d$ and $\rho\boldsymbol{Q}_d$, the feasible region will also change. Afterwards, the MI-GAN is trained based on the changed feasible region to obtain new solutions. If the proposed method is effective, the objective function value calculated by the new output solution should still converge under each experiment.

To fully show the capability of MI-GAN, two cases to change the actual and reactive power loads are considered as follows:

i) Increasing $\boldsymbol{P}_d$ and $\boldsymbol{Q}_d$ by selecting ρ among the set $\{1.1, 1.2, 1.3, 1.4, 1.5\}$.

ii) Decreasing $\boldsymbol{P}_d$ and $\boldsymbol{Q}_d$ by selecting ρ among the set $\{0.9, 0.8, 0.7, 0.6, 0.5\}$.

Two violin plots of the objective function values obtained under different ρ are shown in Fig. 5(a)-(b). Specifically, Fig. 5(a) shows the objective function values when ρ is increasing in the first case, and Fig. 5(b) shows the objective function values when ρ is decreasing in the second case. The size of violins does not show any significant increasing or decreasing patterns in either case. Thus, it could be demonstrated that the changing of net loads has no effect on the stability of the generated solutions by the proposed method.

To better show the moving pattern of the objective function values when ρ changes, the average objective function values and the standard deviations are shown in Table I. It is clearly shown that as ρ increases, the moving percentages of objective function values will also increase. This is because the power generation will increase to fit the loads, which leads to the cost increment. In addition, for case 2, as shown in Table I, the objective function values are fluctuating. Overall, it shows that the proposed method is not sensitive to the dynamic changes in the optimization model.

Besides, according to Table I, the standard deviations of the proposed method are also fluctuating for both increasing and decreasing loads. Such similar standard deviations also reveal that the proposed method is relatively stable when the loads change. Hence, the experiments conducted under uncertainties demonstrate that even if the coefficients of the optimization

model may change, the well-trained MI-GAN could still be re-trained to find the near-optimal solutions effectively.

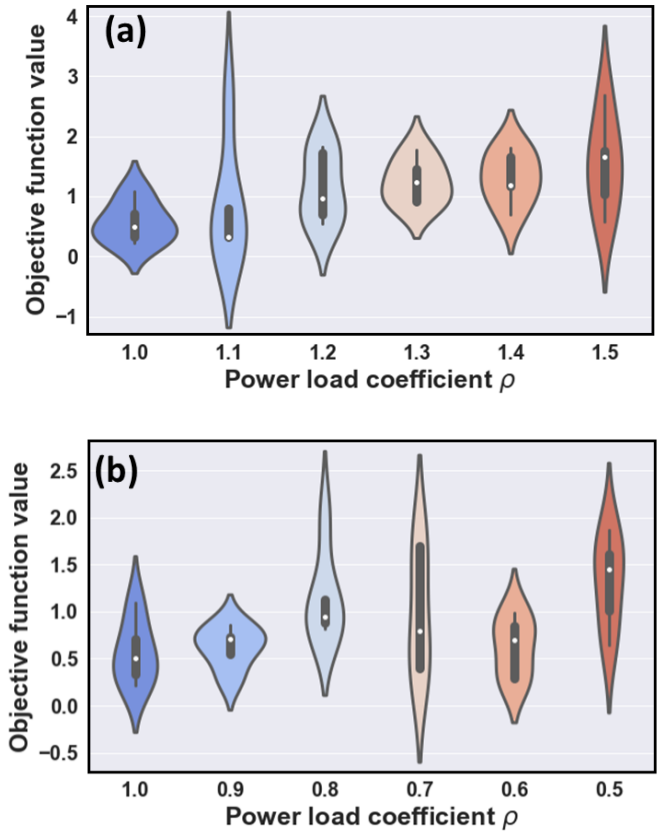

Fig. 5. The violin plot of the objective function value calculated by the proposed method when changing the power loads: (a) The objective function values when ρ is increasing and larger than 1; (b) The objective function values when ρ is decreasing and smaller than 1.

TABLE I. THE PERFORMANCE OF THE PROPOSED MI-GAN BASED FRAMEWORK IN TERMS OF OBJECTIVE FUNCTION VALUES BY AVERAGE (STD)

	ρ				
Case 1	**1.1**	**1.2**	**1.3**	**1.4**	**1.5**
	0.87 (0.90)	1.15 (0.53)	1.25 (0.34)	1.30 (0.40)	1.54 (0.72)
	ρ				
Case 2	**0.9**	**0.8**	**0.7**	**0.6**	**0.5**
	0.63 (0.20)	1.15 (0.44)	0.99 (0.60)	0.62 (0.29)	1.32 (0.44)

IV. CONCLUSIONS

In this paper, a novel MI-GAN based framework is proposed to solve complex non-linear non-convex optimization problems. Compared with the conventional optimization algorithms, the proposed MI-GAN based approach has two main contributions: (i) two model-informed layers, i.e., feasibility filter layer and optimality filter layer, are developed to guarantee the feasibility and improve the optimality of the generated solutions; and (ii) an efficient MI selector is proposed, which incorporates the two new layers, to identify the generated solutions in the GAN-based architecture. In addition, the proposed MI-GAN is compatible with the majority of popular GAN architectures. Therefore, the proposed method could solve various optimization problems by applying different GAN architectures.

The superior performance of the proposed method is demonstrated by one AC-OPF case from IEEE test systems. The solution generated by the proposed method is demonstrated to show its effectiveness. In addition, the recorded running times also demonstrate the computational efficiency of the proposed method. Furthermore, to validate the robustness of the proposed method, the experiments are also conducted when the actual and reactive power loads are not fixed. In such cases, it also shows that the proposed method is effective to handle the uncertainties. Therefore, the experiments in this work have shown the highly potential of the proposed method to be applied in the large-scale nonlinear optimization problems.

In our future work, the output solutions of the proposed MI-GAN based approach will also be compared with the actual optimal solutions to better demonstrate the effectiveness of the proposed method. In addition, the proposed method will also be applied to test more optimization problems, e.g., other AC-OPF cases from IEEE test systems. Furthermore, the recursive iteration algorithm will also be tested under the optimization cases where the historical solutions are not available.

ACKNOWLEDGEMENT

This work was supported in part by the National Science Foundation (NSF) under the Award Number IIP-2141184 and ECCS-2046243.

REFERENCES

[1] Yong, T., & Lasseter, R. H. (2000, July). Stochastic optimal power flow: formulation and solution. In *2000 Power Engineering Society Summer Meeting (Cat. No. 00CH37134)* (Vol. 1, pp. 237-242). IEEE.

[2] Delage, E., & Ye, Y. (2010). Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations research*, *58*(3), 595-612.

[3] Kimball, L. M., Clements, K. A., Pajic, S., & Davis, P. W. (2003, June). Stochastic OPF by constraint relaxation. In *2003 IEEE Bologna Power Tech Conference Proceedings*, (Vol. 4, pp. 5-pp). IEEE.

[4] Martinez-Mares, A., & Fuerte-Esquivel, C. R. (2013). A robust optimization approach for the interdependency analysis of integrated energy systems considering wind power uncertainty. *IEEE Transactions on Power Systems*, *28*(4), 3964-3976.

[5] Capitanescu, F. (2016). Critical review of recent advances and further developments needed in AC optimal power flow. *Electric Power Systems Research*, *136*, 57-68.

[6] Yang, K., Gao, W., & Fan, R. (2021, November). Optimal Power Flow Estimation Using One-Dimensional Convolutional Neural Network. In *2021 North American Power Symposium (NAPS)* (pp. 1-6). IEEE.

[7] Pan, X., Chen, M., Zhao, T., & Low, S. H. (2020). Deepopf: A feasibility-optimized deep neural network approach for ac optimal power flow problems. *arXiv preprint arXiv:2007.01002*.

[8] Schuetz, M. J., Brubaker, J. K., & Katzgraber, H. G. (2022). Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, *4*(4), 367-377.

[9] Li, Y., Zhao, C., & Liu, C. (2022). Model-Informed Generative Adversarial Network (MI-GAN) for Learning Optimal Power Flow. *arXiv preprint arXiv:2206.01864*.

[10] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672-2680).

[11] Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein gan. arXiv 2017. *arXiv preprint arXiv:1701.07875*, *30*, 4.

[12] Yoon, J., Jordon, J., & Schaar, M. (2018, July). Gain: Missing data imputation using generative adversarial nets. In *International conference on machine learning* (pp. 5689-5698). PMLR.

[13] Li, Y., Shi, Z., Liu, C., Tian, W., Kong, Z., & Williams, C. B. (2021). Augmented time regularized generative adversarial network (atr-gan) for data augmentation in online process anomaly detection. *IEEE Transactions on Automation Science and Engineering,* vol. 19, no. 4, pp. 3338-3355.

[14] Liu, Y., Zhou, Y., Liu, X., Dong, F., Wang, C., & Wang, Z., "Wasserstein GAN-based small-sample augmentation for new-generation artificial intelligence: A case study of cancer-staging data in biology," *Engineering*, *5*(1), 156-163, 2019.

[15] Venzke, A., Qu, G., Low, S., & Chatzivasileiadis, S. (2020, November). Learning optimal power flow: Worst-case guarantees for neural networks. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)* (pp. 1-7). IEEE.

[16] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). {TensorFlow}: A System for {Large-Scale} Machine Learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (pp. 265-283).