Received 19 May 2023, accepted 2 June 2023, date of publication 9 June 2023, date of current version 20 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3285199

RESEARCH ARTICLE

Learning to Solve Optimization Problems With Hard Linear Constraints

MEIYI LI[1], SOHEIL KOLOURI[2], (Senior Member, IEEE), AND JAVAD MOHAMMADI[1], (Senior Member, IEEE)

[1]Department of Civil, Architectural and Environmental Engineering, The University of Texas at Austin, Austin, TX 78712

[2]Department of Computer Science, Vanderbilt University, Nashville, TN 37212, USA

Corresponding author: Javad Mohammadi (javadm@utexas.edu)

ABSTRACT Constrained optimization problems have appeared in a wide variety of challenging real-world problems, where constraints often capture the physics of the underlying system. Classic methods for solving these problems relied on iterative algorithms that explored the feasible domain in the search for the best solution. These iterative methods often became the computational bottleneck in decision-making and adversely impacted time-sensitive applications. Recently, neural approximators have shown promise as a replacement for the iterative solvers that can output the optimal solution in a single feed-forward providing rapid solutions to optimization problems. However, enforcing constraints through neural networks remains an open challenge. In this paper, we have developed a neural approximator that maps the inputs to an optimization problem with hard linear constraints to a feasible solution that is nearly optimal. Our proposed approach consists of five main steps: 1) reducing the original problem to optimization on a set of independent variables, 2) finding a gauge function that maps the ℓ_∞-norm unit ball to the feasible set of the reduced problem, 3) learning a neural approximator that maps the optimization's inputs to a virtual optimal point in the ℓ_∞-norm unit ball, and 4) gauge mapping to project the virtual optimal point in the ℓ_∞-norm unit ball onto the feasible space, then 5) finding the values of the dependent variables from the independent variable to recover the solution to the original problem. We can guarantee hard feasibility through this sequence of steps. Unlike the current learning-assisted solutions, our method is free of parameter-tuning (compared to penalty-based methods) and removes iterations altogether. We have demonstrated the performance of our proposed method in quadratic programming in the context of optimal power dispatch (critical to the resiliency of our electric grid) and constrained non-convex optimization in the context of image registration problems. Our results have supported our theoretical findings and demonstrate superior performance in terms of computational time, optimality, and the feasibility of the solution compared to existing approaches.

INDEX TERMS Learning to optimize, hard constraints, machine learning, optimal power flow, image registration.

I. INTRODUCTION

A. MOTIVATION

Constrained optimization problems have been prevalent in computer sciences and engineering, appearing in various applications. Today's optimization solvers employ iterative solvers that primarily leverage first- and second-order techniques (such as (sub)gradient ascent/descent, conjugate gradients, and simplex basis updating methods) to find the optimal solution. These algorithms often provide theoretical convergence guarantees, which is desirable. However, the iterative nature of these solutions increases calculation time and limits their applicability in time-sensitive applications. Many practical setups require solving instances of the same problem repeatedly. Another drawback of existing solutions is that their performance does not improve regardless of how often they deal with the same or similar problems. Furthermore, the availability of algorithms to handle constrained

The associate editor coordinating the review of this manuscript and approving it for publication was Branislav Hredzak.

optimization problems is highly dependent on problem structure, which ranges from problems that can be solved quickly, i.e., linear programming, to problems that have yet to be solved efficiently, e.g., non-convex problems.

The recent interest in using machine learning to improve the efficiency of optimization procedures is fueled by the potential to overcome the discussed shortcomings. Leveraging neural networks can speed up the search process and reduce the number of iterations required to find optimal solutions. The performance of neural approximators can also continually improve as they face more optimization problems. Such neural approximators could enjoy from recent advances in deep learning, including transfer learning, continuous learning and meta learning, etc.

B. RELATED WORKS

1) LEVERAGING DEEP LEARNING TO IMPROVE THE OPTIMIZATION PROCESS OF UNCONSTRAINED PROBLEMS

One of the classical applications of machine learning in optimization has been predicting hyper-parameters, e.g., learning rate [1], momentum decay [2], Lagrangian multipliers [3], etc., to enhance the optimization process. Learning-to-Optimize (L2O) approaches went further by automating the design of optimizers by data-driven approaches [4]. The L2O approach unrolled iterative optimization algorithms and parameterized them [5], [6]. While L2O approaches considerably reduced the total number of iterations required to solve an optimization process, they fell short of eliminating iterations altogether.

Many recent works have focused on replacing the optimization algorithm with a parametric function that directly maps the optimization's input data to the optimal parameters [7]. For instance, the Learning to Optimize the Optimization Process (LOOP) method proposed by [8] showcased promising results by removing iterations and optimizing the optimization process over time and through different problems. While these iteration-free methods could handle a wide range of unconstrained optimization problems, they often struggled with constrained problems. Put differently, today's neural approximators (such as the method proposed by [8]) are effective in finding high-quality (near-optimal) solutions but have limited capabilities in finding feasible solutions.

2) USING PENALTY TERMS TO HANDLE CONSTRAINED OPTIMIZATION PROBLEMS

Incorporating a penalty term to constrain the output of neural approximators is an intuitive strategy. Often the ℓ_2-norm term enforces equality constraints while penalizing square-of-maximum violation deals with inequality constraints [8], [9], [10], [11]. Alternative penalty terms include (i) the difference between the output and its projection on the constraint set [12], (ii) the discrepancy between the output and its projection on a ball centered at the optimal solution [13], (iii) the status deviation of inequality constraints (whether the optimal solution satisfies the inequality constraints) [14], [15], and (iv) the violation of Karush-Kuhn-Tucker (KKT) conditions [16], [17].

One of the main drawbacks of enforcing constraints through penalty terms is the need for parameter tuning. The weights of penalty terms are usually determined heuristically, and the performance of these methods is highly sensitive to these parameters. To address this challenge, [9] presented a design method for proper penalty parameter selection, considering the scalability of the problem. References [18], [19], and [20] combined the Lagrangian dual approaches and deep learning to solve the optimal power flow problems with constraints. Compared to methods with user-selected penalty parameters, the Lagrangian dual approach automatically modifies penalty settings during training and produces more reliable results.

3) DEEP LEARNING FOR OPTIMIZATION WITH HARD BOUNDARIES

Penalty approaches provide a soft boundary on the output because infeasibility is merely punished rather than eliminated. These methods require a trade-off between optimality and feasibility, with the worst-case scenario being that neither is fulfilled. Approximating solutions to optimization problems with hard restrictions were also explored in several recent works. Some works adopted the "projected output" [21] to ensure feasibility. Reference [13] have developed an iterative strategy for modifying the objective function to match model predictions more closely. They use an external solver to maintain feasibility. However, [22] showed that projection-based methods might not provide enough information to find the optimal solution since only a limited number of points on the boundary are accessible (due to projection).

Other methods restructure neural networks to ensure feasibility. For example, the double description approach was employed to cope with linear inequality constraints in [22]. The algorithm constructed a polyhedron-like feasible set iteratively, then optimized over this constructed feasible set. Reference [23] developed a solver for similar equation-constrained problems based on the homotopy continuation method. Reference [24] utilized a correction process to solve the feasible problem using gradient descent in each step. Reference [25] proposed to use alternating projection methods to approximate projections. While these methods satisfy hard-inequality constraints, they require an iterative process for training and testing, which goes against the goal of using deep learning models to replace iterative optimization progress.

References [9] and [24] employed the notion of variable elimination to impose equality constraints when only a subset of the variables is generated in the procedures. The remaining variables can be inferred using the equality constraints. The variable elimination method is iteration free; hence, it directly produces a feasible solution with respect to equality constraints each time an action is executed.

C. CHALLENGES

This paper focuses on addressing the following shortcomings;

- Inefficiency of traditional solvers: Traditional solvers are known to be time-consuming and lack the ability to improve with repeated problem-solving attempts. Additionally, the availability of efficient algorithms for addressing constrained optimization problems, especially non-convex ones, remains a concern.
- Dealing with constrained problems: Neural approximators have demonstrated promising performance in handling unconstrained problems; however, they still struggle to ensure solution feasibility for constrained problems.
- Limitations of penalty terms: The widely-used penalty-based methods merely offer soft boundaries for constraints, and their performance is highly sensitive to the fine-tuning of parameters.
- Iterative nature of hard boundary methods: In existing works, ensuring feasibility with hard boundaries often necessitates iterative processes, which contradicts the goal of developing an iteration-free solution through deep learning models.

D. CONTRIBUTIONS

This paper has developed a trainable parametric function that directly maps problem's input to a high-quality feasible solution of linearly constrained optimization problems. In what follows, we outline our key contributions:

- Rather than solving the original linearly-constrained optimization problem directly, we reformulate and relax it to an equivalent optimization problem in the ℓ_∞-norm unit ball and train a neural network to find an optimal solution in the ℓ_∞-norm unit ball space.
- Inspired by [26], we construct a one-to-one mapping to transfer the optimal solution from the ℓ_∞-norm unit ball to the feasible space of the original constraint set.
- The proposed method outputs near-optimal solutions with feasibility guarantees for equality and inequality constraints of the original linearly-constrained problem.
- We replace iterative optimizers for linear-constrained problems with an iteration-free single-shot neural solver.
- The proposed method fundamentally differs from existing learning-based approaches, such as penalizing constraint violations and employing restoration techniques to obtain feasible solutions. Unlike these methods, the proposed approach does not require a separate restoration process.

II. PROBLEM FORMULATION

In this section, we introduce the notations and problem formulations. Let us consider the following linear-constraint optimization problem:

$$\min \mathrm{f}(\mathbf{u}, \mathbf{x}) \tag{1a}$$

$$\texttt{s.t.} \quad \mathbf{A}_{\texttt{eq}}\mathbf{u} + \mathbf{B}_{\texttt{eq}}\mathbf{x} + \mathbf{b}_{\texttt{eq}} = \mathbf{0} \tag{1b}$$

$$\mathbf{A}_{\texttt{ineq}}\mathbf{u} + \mathbf{B}_{\texttt{ineq}}\mathbf{x} + \mathbf{b}_{\texttt{ineq}} \leq \mathbf{0} \tag{1c}$$

Here, $\mathbf{u} \in \mathbb{R}^{N_{\texttt{opt}}}$ denotes the vector of optimization variables, whereas $\mathbf{x} \in \mathbb{R}^{N_{\texttt{inp}}}$ represents the input vector. Also, $N_{\texttt{opt}}$ and $N_{\texttt{inp}}$ represent dimensions of optimization variables and input vectors. $\mathbf{A}_{\texttt{eq}} \in \mathbb{R}^{N_{\texttt{eq}} \times N_{\texttt{opt}}}$, $\mathbf{B}_{\texttt{eq}} \in \mathbb{R}^{N_{\texttt{eq}} \times N_{\texttt{inp}}}$, and $\mathbf{b}_{\texttt{eq}} \in \mathbb{R}^{N_{\texttt{eq}}}$ are parameters for equality constraints, whereas $\mathbf{A}_{\texttt{ineq}} \in \mathbb{R}^{N_{\texttt{ineq}} \times N_{\texttt{opt}}}$, $\mathbf{B}_{\texttt{ineq}} \in \mathbb{R}^{N_{\texttt{ineq}} \times N_{\texttt{inp}}}$, and $\mathbf{b}_{\texttt{ineq}} \in \mathbb{R}^{N_{\texttt{ineq}}}$ define inequality constraints. Also, $N_{\texttt{ineq}}$ and $N_{\texttt{eq}}$ refer to dimensions of inequality and equality constraints. Furthermore, equations (1b) and (1c) establish element-wise equality and inequality relations with a zero vector (i.e., $\mathbf{0}$). Moreover, $\mathrm{f}(\mathbf{u}, \mathbf{x})$ refers to any convex or non-convex objective function. We assume that problem (1) is an under-determined problem, i.e., $\mathrm{rank}(\mathbf{A}_{\texttt{eq}}) = N_{\texttt{eq}} < N_{\texttt{opt}}$. To simplify the notation, we will refer to the optimal solution and constraint set of problem (1) as $\mathbf{u}^*$ and $\mathcal{S}$, respectively. $\mathcal{S}$ is non-empty. Given that $\mathbf{u}$ is bounded in most practical settings, $\mathcal{S}$ is considered a bounded set. Therefore problem (1) can be presented by the following abstract form,

$$\min \mathrm{f}(\mathbf{u}, \mathbf{x}) \quad \texttt{s.t.} \quad \mathbf{u} \in \mathcal{S}(\mathbf{x}) \tag{2}$$

Similar to the framework proposed by [8], we replace the classic iterative solvers with a trainable parametric function ξ_θ that directly maps the input of the optimization problem to the optimal parameters in a single feed-forward. By bypassing the traditional iterative solutions, the method overcomes one of the significant optimization bottlenecks enabling near real-time optimization in a wide range of critical applications.

Note, problem (1) is a constrained optimization problem. Hence, the feasibility of ξ_θ's output should be ensured. It is straightforward to use activation functions to guarantee feasibility when $\mathcal{S}$ constitutes an ℓ_∞-norm ball. However, it is challenging to utilize neural approximators to solve (2) where $\mathcal{S}$ includes coupled constraints and variables. This paper extends the prior works (e.g., [5], [8]) for solving unconstrained problems to solve linearly constrained optimization problems (referred to as $\mathcal{LOOP}-\mathcal{LC}$, Learning to Optimize the Optimization Process with Linear Constraints). In what follows, we will first introduce the basics of $\mathcal{LOOP}-\mathcal{LC}$ method and provide theoretical guarantees to justify the feasibility of the resulting solution. Later, We will showcase the performance of $\mathcal{LOOP}-\mathcal{LC}$ in quadratic programming in the context of the optimal power dispatch (critical to the resiliency of our electric grid) and a constrained non-convex optimization in the context of image registration problems.

III. PROPOSED METHOD

A. PROPOSED ARCHITECTURE

The rest of this subsection is dedicated to presenting an abstract overview of our proposed framework, $\mathcal{LOOP}-\mathcal{LC}$.

1) OPTIMIZATION REFORMULATION

We will adopt a variable elimination technique [24] to reformulate (1) as a reduced-dimension optimization problem

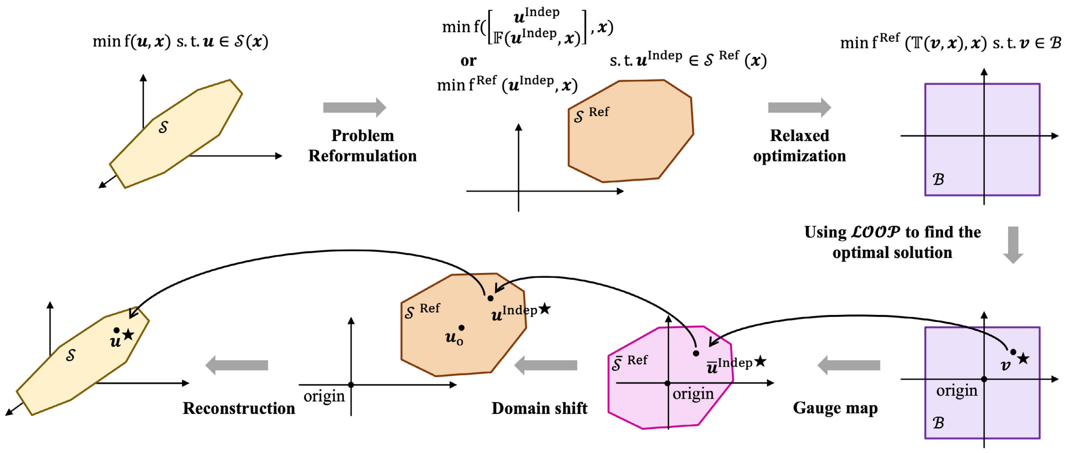

FIGURE 1. Structure of the proposed $\mathcal{LOOP}-\mathcal{LC}$ model.

with only inequalities. This approach first decomposes the set of optimization variables $\mathbf{u}$ to independent ($\mathbf{u}^{\mathtt{Indep}}$) and dependent ($\mathbf{u}^{\mathtt{Dep}}$) parts. Then uses equality constraints to find relationships between $\mathbf{u}^{\mathtt{Indep}}$ and $\mathbf{u}^{\mathtt{Dep}}$, i.e., $\mathbf{u}^{\mathtt{Dep}} = \mathbb{F}(\mathbf{u}^{\mathtt{Indep}}, \mathbf{x})$. The reformulated problem will be referred to as,

$$\min \mathrm{f}^{\mathtt{Ref}}(\mathbf{u}^{\mathtt{Indep}}, \mathbf{x}) \quad \mathtt{s.t.} \quad \mathbf{u}^{\mathtt{Indep}} \in \mathcal{S}^{\mathtt{Ref}}(\mathbf{x}) \tag{3}$$

where $\mathrm{f}^{\mathtt{Ref}}$ and $\mathcal{S}^{\mathtt{Ref}}$ are reformulation of f and $\mathcal{S}$ that only depends on $\mathbf{u}^{\mathtt{Indep}}$, respectively.

2) UTILIZING NEURAL NETWORK TO OPTIMIZE (3)

Instead of solving the original problem directly, we train a neural network to find the optimal solution $\mathbf{v}$ in the ℓ_∞-norm unit ball, $\mathcal{B}$.

$$\min \mathrm{f}^{\mathtt{Ref}}(\mathbb{T}(\mathbf{v}, \mathbf{x}), \mathbf{x}) \quad \mathtt{s.t.} \quad \mathbf{v} \in \mathcal{B} \tag{4}$$

Layers of the neural network will ensure that the resulting $\mathbf{v}^\star$ will stay within $\mathcal{B}$. Later we will provide a one-to-one mapping to transfer the resulting solutions from ℓ_∞-norm unit ball to the feasible space of constraint set $\mathcal{S}^{\mathtt{Ref}}$. This mapping will be denoted as $\mathbf{u}^{\mathtt{Indep}} = \mathbb{T}(\mathbf{v}, \mathbf{x})$.

3) ENSURING FEASIBILITY BY FINDING $\mathbb{T}$

We use the gauge map [26] to build a one-to-one mapping between $\mathcal{B}$ (i.e., ℓ_∞-norm unit ball) and $\mathcal{S}^{\mathtt{Ref}}$ (i.e., feasible domain of the reformulated problem) spaces. The gauge map requires the destination space to encompass the origin as an interior point [27]. Thus, rather than directly mapping the ℓ_∞-norm unit ball into the desired feasible domain $\mathcal{S}^{\mathtt{Ref}}$, we first shift the desired domain by one of its interior points $\mathbf{u}_\circ$ to construct an "intermediate domain" $\bar{\mathcal{S}}^{\mathtt{Ref}}$. The intermediate domain shares the same geometric properties with $\mathcal{S}^{\mathtt{Ref}}$ but contains the origin as an interior point. In a nutshell, this process enables leveraging the gauge function to map the ℓ_∞-norm unit ball into an "intermediate domain" and then shift the intermediate domain to the desired domain, $\mathcal{S}^{\mathtt{Ref}}$.

4) EQUALITY COMPLETION

The previous step tackles the reduced-dimension optimization problem (i.e., formulation (3)), hence, it finds optimal values for a subset of optimization variables (i.e., $\mathbf{u}^{\mathtt{Indep}\star}$). The remaining variables $\mathbf{u}^{\mathtt{Dep}\star}$, will be determined by using the algebraic relationship $\mathbb{F}$ between optimization variables that are given by equality constraints of (2).

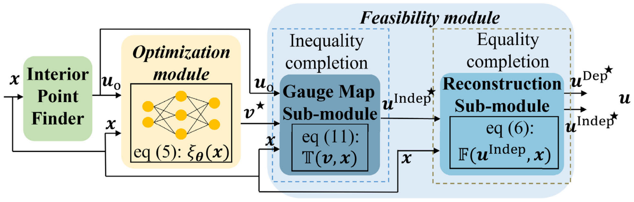

FIGURE 2. The $\mathcal{LOOP}-\mathcal{LC}$ framework is composed of interior point finder, optimization, and feasibility modules.

The forthcoming subsections present the details of the steps presented in this subsection. The modular architecture of $\mathcal{LOOP}-\mathcal{LC}$, depicted in Figure 2, ensures achieving a high-quality (near-optimal) feasible solution.

B. OPTIMIZATION MODULE

As stated in section III-A, we incorporate a neural network to learn a high-quality solution to problem (4). Note that the gauge map sub-module (that will be introduced later) takes in the interior point $\mathbf{u}_\circ (\in \mathcal{S}^{\mathtt{Ref}})$, hence the **input**s of the optimality module include both $\mathbf{u}_\circ$ and $\mathbf{x}$. Let $\boldsymbol{\theta}$ denote the weights of the neural network. The **output** of the optimality module is prediction $\mathbf{v}$ that lies in the ℓ_∞-norm unit ball (i.e., $\mathbf{v}^\star$):

$$\mathbf{v}^\star = \xi_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}_\circ) \tag{5}$$

Choosing the proper activation functions, e.g., the hyperbolic tangent function, ensures that the resulting $\mathbf{v}$ stays in the feasible range of ℓ_∞-norm unit ball, i.e., $[-1, 1]$. Later, $\mathbf{v}^\star$ will pass through the feasibility module to generate $\mathbf{u}^\star$.

The optimality module uses two training approaches, as illustrated in Figure 3: 1) with a solver in the loop, and 2) without a solver in the loop, i.e., directly minimizing the objective function. Assume there are N input data points indexed as $\mathbf{x}^{(i)}$, the respective output denoted as $\mathbf{u}^{(i)}$. The loss function with a solver in the loop is a discrepancy/distance function $\mathrm{d} : \mathbb{R}^{N_{\text{opt}}} \times \mathbb{R}^{N_{\text{opt}}} \rightarrow \mathbb{R}_+$ defined in $\mathbb{R}^{N_{\text{opt}}}$ and compares the difference between the output of the $\mathcal{LOOP}-\mathcal{LC}$ model and the optimal solution calculated using commercial solvers, i.e. $L = \sum_{i=1}^{N} \mathrm{d}(\mathbf{u}^{(i)}, \mathbf{u}^{(i)*})$. Without a solver, the loss function is just the expected value of the objective function, i.e., $L = \sum_{i=1}^{N} \mathrm{f}(\mathbf{u}^{(i)}, \mathbf{x}^{(i)})$.

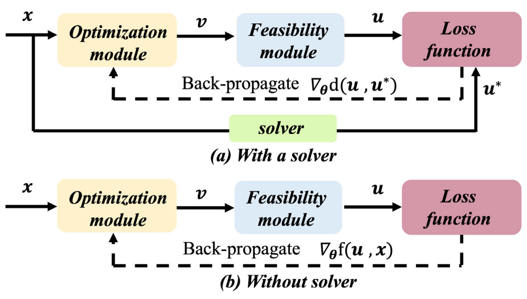

FIGURE 3. Two training approaches of $\mathcal{LOOP}-\mathcal{LC}$ model.

C. FEASIBILITY MODULE

While the optimality module subsection provides a high-quality solution, it does not necessarily produce a feasible solution. To this end, the feasibility module will first map $\mathbf{v}^\star$ onto the desired feasible domain and then compute a full-dimensional output $\mathbf{u}^\star$. In this subsection, we will present how this module enforces feasibility through equality completion and inequality satisfaction.

1) EQUALITY COMPLETION

This sub-module reconstructs the equality equations of problem (1). We first divide the elements in $\mathbf{u}$ into two groups: $(N_{\text{opt}} - N_{\text{eq}})$ independent parameters and N_{eq} dependent parameters. Dependent parameters $\mathbf{u}^{\texttt{Dep}} \in \mathbb{R}^{N_{\text{eq}}}$ are defined by all the equality constraints in problem (1), whereas independent parameters $\mathbf{u}^{\texttt{Indep}} \in \mathbb{R}^{(N_{\text{opt}}-N_{\text{eq}})}$.

a: CLAIM

Let us define function $\mathbb{F}$ as $\mathbb{F} : \mathbb{R}^{(N_{\text{opt}}-N_{\text{eq}})} \rightarrow \mathbb{R}^{N_{\text{eq}}}$ s.t. $\mathbf{u}^{\texttt{Dep}} = \mathbb{F}(\mathbf{u}^{\texttt{Indep}}, \mathbf{x})$ where $\mathbf{A}_{\text{eq}}[\mathbf{u}^{\texttt{Indep}^{\mathrm{T}}}, \mathbf{u}^{\texttt{Dep}^{\mathrm{T}}}]^{\mathrm{T}} = -\mathbf{B}_{\text{eq}}\mathbf{x} - \mathbf{b}_{\text{eq}}$. Then, there exists such an $\mathbb{F}$ for a linear-constraint set.

b: JUSTIFICATION

Let us select N_{eq} linearly independent columns in $\mathbf{A}_{\text{eq}}$ and group them into $\mathbf{A}_{\text{eq}}^{\texttt{Dep}}$. The other columns form $\mathbf{A}_{\text{eq}}^{\texttt{Indep}}$. Then we have $\mathbf{A}_{\text{eq}}^{\texttt{Indep}}\mathbf{u}^{\texttt{Indep}} + \mathbf{A}_{\text{eq}}^{\texttt{Dep}}\mathbf{u}^{\texttt{Dep}} + \mathbf{B}_{\text{eq}}\mathbf{x} + \mathbf{b}_{\text{eq}} = \mathbf{0}$. That is:

$$\begin{aligned}\mathbf{u}^{\texttt{Dep}} &= \mathbb{F}(\mathbf{u}^{\texttt{Indep}}, \mathbf{x}) \\ &= -\mathbf{A}_{\text{eq}}^{\texttt{Dep}^{-1}}\mathbf{A}_{\text{eq}}^{\texttt{Indep}}\mathbf{u}^{\texttt{Indep}} - \mathbf{A}_{\text{eq}}^{\texttt{Dep}^{-1}}(\mathbf{B}_{\text{eq}}\mathbf{x} + \mathbf{b}_{\text{eq}})\end{aligned} \tag{6}$$

By incorporating reconstruction function $\mathbb{F}$, problem (1) changes to minimizing $\mathrm{f}([\mathbf{u}^{\texttt{Indep}^{\mathrm{T}}}, \mathbb{F}(\mathbf{u}^{\texttt{Indep}}, \mathbf{x})^{\mathrm{T}}]^{\mathrm{T}}, \mathbf{x})$ with merely inequality constraints $\mathbf{A}_{\text{ineq}}[\mathbf{u}^{\texttt{Indep}^{\mathrm{T}}}, \mathbb{F}(\mathbf{u}^{\texttt{Indep}}, \mathbf{x})^{\mathrm{T}}]^{\mathrm{T}} + \mathbf{B}_{\text{ineq}}\mathbf{x} + \mathbf{b}_{\text{ineq}} \leq \mathbf{0}$.

We rewrite $\mathbf{A}_{\text{ineq}}$ as $\begin{bmatrix}\mathbf{A}_{\text{ineq}}^{\texttt{Indep}} & \mathbf{A}_{\text{ineq}}^{\texttt{Dep}}\end{bmatrix}$ in accordance with $\mathbf{u}^{\texttt{Indep}}$ and $\mathbf{u}^{\texttt{Dep}}$. Then, according to (6) and (1c), the inequality constraints can be written as: $(\mathbf{A}_{\text{ineq}}^{\texttt{Indep}} - \mathbf{A}_{\text{ineq}}^{\texttt{Dep}}\mathbf{A}_{\text{eq}}^{\texttt{Dep}^{-1}}\mathbf{A}_{\text{eq}}^{\texttt{Indep}})\mathbf{u}^{\texttt{Indep}} - \mathbf{A}_{\text{ineq}}^{\texttt{Dep}}\mathbf{A}_{\text{eq}}^{\texttt{Dep}^{-1}}(\mathbf{B}_{\text{eq}}\mathbf{x} + \mathbf{b}_{\text{eq}}) + \mathbf{B}_{\text{ineq}}\mathbf{x} + \mathbf{b}_{\text{ineq}} \leq \mathbf{0}$. As the next step, let us rewrite problem (1) as a reduced-dimension optimization problem with only inequalities, as (3). This means that $\mathbf{A} = \mathbf{A}_{\text{ineq}}^{\texttt{Indep}} - \mathbf{A}_{\text{ineq}}^{\texttt{Dep}}\mathbf{A}_{\text{eq}}^{\texttt{Dep}^{-1}}\mathbf{A}_{\text{eq}}^{\texttt{Indep}}$, $\mathbf{B} = \mathbf{B}_{\text{ineq}} - \mathbf{A}_{\text{ineq}}^{\texttt{Dep}}\mathbf{A}_{\text{eq}}^{\texttt{Dep}^{-1}}\mathbf{B}_{\text{eq}}$, $\mathbf{b} = \mathbf{b}_{\text{ineq}} - \mathbf{A}_{\text{ineq}}^{\texttt{Dep}}\mathbf{A}_{\text{eq}}^{\texttt{Dep}^{-1}}\mathbf{b}_{\text{eq}}$. That is:

$$\mathcal{S}^{\text{Ref}} = \left\{\mathbf{u}^{\texttt{Indep}} | \mathbf{A}\mathbf{u}^{\texttt{Indep}} + \mathbf{B}\mathbf{x} + \mathbf{b} \leq \mathbf{0}\right\} \tag{7}$$

$$\mathrm{f}^{\text{Ref}}(\mathbf{u}^{\texttt{Indep}}, \mathbf{x}) = \mathrm{f}(\begin{bmatrix}\mathbf{u}^{\texttt{Indep}} \\ \mathbb{F}(\mathbf{u}^{\texttt{Indep}}, \mathbf{x})\end{bmatrix}, \mathbf{x}) \tag{8}$$

Thus, we can solve (3) and later reconstruct the remaining parameters according to (6). Therefore, the reconstruction sub-module takes in independent parameters $\mathbf{u}^{\texttt{Indep}}$ and $\mathbf{x}$ and outputs $\mathbf{u}^{\texttt{Dep}}$. The existence of $\mathbb{F}$ guarantees that the resulting optimization solution $\mathbf{u}^\star = [\mathbf{u}^{\texttt{Indep}\star\mathrm{T}}, \mathbf{u}^{\texttt{Dep}\star\mathrm{T}}]^{\mathrm{T}}$ satisfy the equality constraints of (1b).

2) INEQUALITY COMPLETION

In order to enforce inequality constraints in (3), we incorporate the gauge map sub-module, which is based on the Minkowski function defined below.

Definition 1 (Minkowski function): Given a convex and compact set $\mathcal{C} \subset \mathbb{R}^n$, assume the origin belongs to the algebraic interior of $\mathcal{C}$ and $\mathbf{c} \in \mathcal{C}$. The Minkowski function associated with $\mathcal{C}$ is defined by $\varphi_{\mathcal{C}}(\mathbf{c}) = \inf\{r > 0 : \mathbf{c} \in r\mathcal{C}\}$.

The space $\mathcal{C}$ is a polytope (encompassing the origin) and is defined as $\mathcal{C} = \left\{\mathbf{c} \in \mathbb{R}^n | \mathbf{H}^j\mathbf{c} \leq h^j, j = 1 \ldots m\right\}$. Here, $\mathbf{H}^j$ is a $1 \times n$ row vector. The Minkowski function associated with $\mathcal{C}$ is defined as:

$$\varphi_{\mathcal{C}}(\mathbf{c}) = \max_j\{\frac{\mathbf{H}^j\mathbf{c}}{h^j}\} \tag{9}$$

The Minkowski function allows "translating" specific geometric properties of a subset to a (particular) algebraic property of another subset. The "translation" is enabled by the gauge map.

Definition 2 (Gauge mapping function): Let us consider two convex and compact sets $\mathcal{C} \subset \mathbb{R}^n$ and $\bar{\mathcal{C}} \subset \mathbb{R}^n$. Let us assume that the origin belongs to the algebraic interior of both

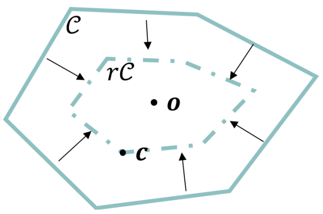

FIGURE 4. The sub-level sets of a Minkowski function are achieved by linearly scaling the set $\mathcal{C}$. Specifically, any point c could be referred to by the distance to the origin $\varphi_{\mathcal{C}}(\mathbf{c})$ and the direction $\mathbf{c}/\varphi_{\mathcal{C}}(\mathbf{c})$.

$\mathcal{C}$ and $\bar{\mathcal{C}}$. The gauge map $G : \mathcal{C} \rightarrow \bar{\mathcal{C}}$ is a bijection function defined as $\bar{\mathbf{c}} = G(\mathbf{c}, \mathcal{C}, \bar{\mathcal{C}}) = \frac{\varphi_{\mathcal{C}}(\mathbf{c})}{\varphi_{\bar{\mathcal{C}}}(\mathbf{c})}\mathbf{c}$. Here, $\mathbf{c} \in \mathcal{C}$ and $\bar{\mathbf{c}} \in \bar{\mathcal{C}}$.

This property means a feasible range with a simple geometric shape (such as ℓ_∞-norm unit ball $\mathcal{B}$) can be translated to a complex feasible range (such as $\bar{\mathcal{S}}^{\texttt{Ref}}$). Since the gauge map function provides a one-to-one mapping, choosing a point in $\mathcal{B}$ is equivalent to choosing a point in $\bar{\mathcal{S}}^{\texttt{Ref}}$.

As shown in Figure 4, the gauge map function is based on the concept of an absorbing set that can be deflated in accordance with the origin. To apply the gauge map function, however, we must temporarily "shift" the desired feasible domain $\mathcal{S}^{\texttt{Ref}}$ by one of its interior points $\mathbf{u}_\circ$ to make it a set $\bar{\mathcal{S}}^{\texttt{Ref}}$ that contains the origin as an interior point. Thus,

$$\bar{\mathcal{S}}^{\texttt{Ref}} = \{\bar{\mathbf{u}}^{\texttt{Indep}} | (\bar{\mathbf{u}}^{\texttt{Indep}} + \mathbf{u}_\circ) \in \mathcal{S}^{\texttt{Ref}}\} \tag{10}$$

The set $\bar{\mathcal{S}}^{\texttt{Ref}}$ serves as a bridge connecting the ℓ_∞-norm unit ball $\mathcal{B}$ and the desired feasible domain $\mathcal{S}^{\texttt{Ref}}$. We use the gauge map to translate $\mathbf{v}^\star$ into a $\bar{\mathbf{u}}^{\texttt{Indep}\star}$ in $\bar{\mathcal{S}}^{\texttt{Ref}}$ and then shift it to $\mathbf{u}^{\texttt{Indep}\star}$ in $\mathcal{S}^{\texttt{Ref}}$. Put differently,

$$\mathbf{u}^{\texttt{Indep}\star} = \mathbb{T}(\mathbf{v}^\star, \mathbf{x}) = \bar{\mathbf{u}}^{\texttt{Indep}\star} + \mathbf{u}_\circ = \frac{\varphi_{\mathcal{B}}(\mathbf{v}^\star)}{\varphi_{\bar{\mathcal{S}}^{\texttt{Ref}}}(\mathbf{v}^\star)}\mathbf{v}^\star + \mathbf{u}_\circ \tag{11}$$

Therefore, the **inputs** of the gauge map sub-module are $\mathbf{x}$, $\mathbf{v}^\star$ and an interior point $\mathbf{u}_\circ$. The **output** is independent parameters $\mathbf{u}^{\texttt{Indep}\star}$.

All in all, given any $\mathbf{v}^\star$ in the ℓ_∞-norm unit ball, the feasibility module first produces a reduced-size solution $\mathbf{u}^{\texttt{Indep}\star}$ and then expands it to a full-dimension solution $\mathbf{u}^\star$. The gauge map and reconstruction functions will enforce both the equality constraints and inequality constraints.

D. INTERIOR POINT FINDER

As discussed in the optimization and feasibility modules, we need to use an interior point in $\mathcal{S}^{\texttt{Ref}}$ to construct the intermediate domain $\bar{\mathcal{S}}^{\texttt{Ref}}$. The difficulty of finding interior points stems from the fact that $\mathcal{S}^{\texttt{Ref}}$ varies as $\mathbf{x}$ changes. Therefore, an interior point of $\mathcal{S}^{\texttt{Ref}}(\mathbf{x}^{(i)})$ may not be an interior point of $\mathcal{S}^{\texttt{Ref}}(\mathbf{x}^{(j)}), j \neq i$. We start by making an assumption that given any $\mathbf{x}^{(i)}$, $i = 1 \ldots m$, $\exists \mathcal{S}^{\texttt{int}} \subset \mathcal{S}^{\texttt{Ref}}(\mathbf{x}^{(i)})$ ([26]). Then any point in $\mathcal{S}^{\texttt{int}}$ is an interior point. This assumption holds when the input $\mathbf{x}$ (or $\mathcal{S}^{\texttt{Ref}}(\mathbf{x})$) is under small disturbances. In this subsection, we present an initial artificial problem method to find out an interior point for more general cases.

Inspired by the implementation of the interior-point method [28], we first define the following problem using the pseudo-variable $u_a \in \mathbb{R}$,

$$\min M u_a \tag{12a}$$
$$\texttt{s.t.} \quad \mathbf{A}\mathbf{u}^{\texttt{Indep}} + \mathbf{B}\mathbf{x} + \mathbf{b} - \mathbf{1}u_a \leq \mathbf{0} \tag{12b}$$

Here, M is a large coefficient. $\mathbf{1}$ is an all-one column vector. The solution to problem (12) is an interior point to $\mathcal{S}^{\texttt{Ref}}$.

Let us note $\left[\mathbf{u}^{\texttt{Indep}\blacklozenge\texttt{T}}, u_a^\blacklozenge\right]^{\texttt{T}}$ as the solution to problem (12). The interior point of $\mathcal{S}^{\texttt{Ref}}$ exists if and only if $u_a^\blacklozenge < 0$. Thus, solving problem (12) yields an interior point $\mathbf{u}^{\texttt{Indep}\blacklozenge} \in \mathcal{S}^{\texttt{Ref}}$ if $u_a^\blacklozenge < 0$.

Note that problem (12) represents a linear programming problem that can be efficiently solved using traditional solvers. Although the development of an interior point finding method is not the primary focus of this paper, we acknowledge the critical role of securing an initial interior point in our proposed method. Importantly, to obtain a feasible point through our neural approximator, we only need to identify a negative $u_a^\blacklozenge$. Consequently, we can approach problem (12) with a more flexible convergence criterion or a relaxed convergence tolerance, thereby significantly reducing the computational time required to find an interior point.

The pseudo-code for $\mathcal{LOOP}-\mathcal{LC}$ model training summarizes the proposed method section.

Algorithm 1 $\mathcal{LOOP}-\mathcal{LC}$Model Training

```
Require: Neural network ξ_θ initialization (architecture, weights, and bias), Hyperparameters (learning rate, number of epochs, batch size, etc.), Data points, Constrained-optimization problem specification, Interior points
1: for epoch in 1 to max_epochs do
2:     Forward propagation:
         -predict virtual prediction $\mathbf{v}$ using current weights, bias according to (5)
         -compute independent variables $\mathbf{u}^{\texttt{Indep}}$ according to (11)
         -produce a full-size prediction $\mathbf{u}$ according to (6)
3:     Compute loss function $L$
4:     Backward propagation: compute gradients $\frac{\partial L}{\partial \boldsymbol{\theta}}$
5:     Update weights and bias
6: end for
```

IV. RESULTS

A. CONVEX PROBLEM: DC OPTIMAL POWER FLOW

The DC optimal power flow (DCOPF) problem [35] minimizes the cost of procuring electricity in a power grid while respecting the system's limitations. The formulation of the DCOPF problem is given below.

$$\min \mathrm{f}(\mathbf{P}_{\mathrm{G}}) = \sum_{g=1}^{N_{\mathrm{G}}} \mathrm{f}^g(P_{\mathrm{G}}^g) \tag{13a}$$

$$s.t. \quad \sum_{g=1}^{N_{\mathrm{G}}} P_{\mathrm{G}}^{g} = \sum_{i=1}^{N_{\mathrm{D}}} P_{\mathrm{D}}^{i}, \underline{\mathbf{P}}_{\mathrm{G}} \leq \mathbf{P}_{\mathrm{G}} \leq \overline{\mathbf{P}}_{\mathrm{G}} \tag{13b}$$

$$\mathbf{D}_{\mathrm{G}}\mathbf{P}_{\mathrm{G}} - \mathbf{D}_{\mathrm{D}}\mathbf{P}_{\mathrm{D}} \leq \mathbf{P}_{\mathtt{line}} \tag{13c}$$

Where $\mathbf{P}_{\mathrm{G}} = [P_{\mathrm{G}}^{1} \dots P_{\mathrm{G}}^{g} \dots P_{\mathrm{G}}^{N_{\mathrm{G}}}]^{T}$ refers to the vector of electric power generation, $\mathbf{P}_{\mathrm{D}} = [P_{\mathrm{D}}^{1} \dots P_{\mathrm{D}}^{i} \dots P_{\mathrm{D}}^{N_{\mathrm{D}}}]^{T}$ denote the vector of electric demands. The equality constraint enforces the balance of supply and demand, while the inequality constraint respects the physics of the electric system. Here, $\mathbf{D}_{\mathrm{g}}$ and $\mathbf{D}_{\mathrm{D}}$ (the so called power transfer distribution factor matrix [29]) capture the physics of the electric network.

1) DATASET

We use the publicly available IEEE 200-bus system data set, available via the MATPOWER [30], as the seed information to generate 200 data points (with a train/test ratio of 1:1). The IEEE 200-bus system is a 200 nodes graph representing a realistic electric grid. This system consists of 200 load nodes and 49 generation nodes. We consider a 10-percentage fluctuation of each load node.

2) COMPARISON

We compare $\mathcal{LOOP}-\mathcal{LC}$ against the three recent learning-based optimization methods; (i) projection [21], (ii) penalty [8], and (iii) DC3 [24] methods, as well as two well-known commercial solvers (i.e., matpower 7.1 [30] and CVXOPT [31]). The projection method projects the output of the neural network onto the feasible range, while the penalty method adds a ℓ_2-norm term to the loss function. Moreover, the DC3 method utilizes the ℓ_2-norm penalty term in the objective function to iteratively enforce the output of the neural networks to satisfy optimization constraints. These methods are illustratively compared in Figure 5.

3) PARAMETERS

We use a fixed neural network architecture for projection, penalty, and DC3 methods: fully connected with one hidden layer of size 16, including the rectified linear unit (ReLU) activation. An extra Tanh activation is added to the output layer for $\mathcal{LOOP}-\mathcal{LC}$ model. Different hyperparameters were tuned to maximize performance for each method individually (see Table 1). Since the DC3 method is based on an inner iteration, the performance improves over the course of iterations. We let it run till a similar time limit as $\mathcal{LOOP}-\mathcal{LC}$ model to facilitate comparison.

4) INTERIOR POINT FINDER

$\mathcal{S}^{\mathtt{Ref}}$ varies as the electric demand changes; therefore, we use the initial artificial problem method (discussed in Section III-D) to find interior points for the $\mathcal{LOOP}-\mathcal{LC}$ method.

5) RESULTS

Based on Table 2, the penalty method achieves the best execution time. However, at its core, this method introduces a trade-off between optimality and feasibility. The convergence speed-up comes at the cost of an increased feasibility gap. The same trade-off manifests itself in the DC3 method, meaning that the feasibility gap decreases with more inner iterations, which may adversely impact optimality and solution time. The performance of both the penalty approach and the DC3 method is sensitive to hyperparameters. Poor choices of hyperparameters may lead to divergence of training or testing (for example, the step size of 0.001 for DC3 results in divergence).

Our results show that only the $\mathcal{LOOP}-\mathcal{LC}$ and projection method satisfies hard feasibility constraints, while $\mathcal{LOOP}-\mathcal{LC}$'s solution time performance surpasses the projection method by a large margin. Specifically, given the interior point, the $\mathcal{LOOP}-\mathcal{LC}$ method will be executed three orders of magnitude faster than the projection method.

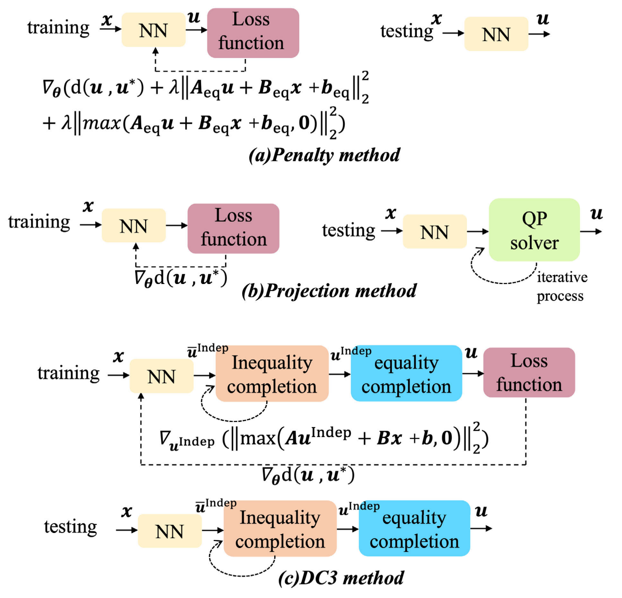
training
x
NN
u
Loss function
testing
x
NN
u
$\nabla_{\theta}(\mathrm{d}(u,u^{*}) + \lambda\|A_{\mathrm{eq}}u + B_{\mathrm{eq}}x + b_{\mathrm{eq}}\|_2^2 + \lambda\|max(A_{\mathrm{eq}}u + B_{\mathrm{eq}}x + b_{\mathrm{eq}}, 0)\|_2^2)$
(a)Penalty method
training
x
NN
Loss function
$\nabla_{\theta}\mathrm{d}(u,u^{*})$
testing
x
NN
QP solver
u
iterative process
(b)Projection method
training
x
NN
$\bar{u}^{\mathrm{Indep}}$
Inequality completion
u^{Indep}
equality completion
u
Loss function
$\nabla_{u^{\mathrm{Indep}}}(\|\max(Au^{\mathrm{Indep}} + Bx + b, 0)\|_2^2)$
$\nabla_{\theta}\mathrm{d}(u,u^{*})$
testing
x
NN
$\bar{u}^{\mathrm{Indep}}$
Inequality completion
u^{Indep}
equality completion
u
(c)DC3 method

FIGURE 5. We present an illustrative comparison between functionalities of projection, penalty, and DC3 method for solving (1).

TABLE 1. Hyperparameters tested for different methods. The final parameter values are identified in bold.

Penalty	penalty coefficient	10^2,10^3,$\mathbf{10^4}$,10^5,10^6
DC3	step size	10^{-2},10^{-3},$\mathbf{10^{-4}}$,10^{-5},10^{-6}
	inner iteration times for testing	1,**3**,5,10
	inner iteration time for training	1,**3**,5,10

B. NONCONVEX PROBLEM: IMAGE REGISTRATION

Image registration is a fundamental image analysis problem that aims to optimize the transform function that moves the coordinate system of one image to another [32]. Let $\mathcal{X} = [-1, 1]^2$ denote the domain of an image, and let us denote $I : \mathcal{X} \to [0, 1]$ as an image defined in this domain. Also, we will refer to the source image as I_{S} and

TABLE 2. This table presents the results of using different methods to solve the DCOPF problem. The time is reported as the average per instance in milliseconds. The Optimality gap is measured as $\frac{1}{N}\sum_{i=1}^{N}\frac{\left\|\mathbf{u}^{(i)}-\mathbf{u}^{(i)*}\right\|_1}{\left\|\mathbf{u}^{(i)*}\right\|_1}$. The Feasibility gap is calculated using $\frac{1}{N}$ $\left(\left\|\max(\mathbf{A}_{\mathtt{ineq}}\mathbf{u}^{(i)}+\mathbf{B}_{\mathtt{ineq}}\mathbf{x}^{(i)}+\mathbf{b}_{\mathtt{ineq}},\mathbf{0})\right\|_1+\left\|\mathbf{A}_{\mathtt{eq}}\mathbf{u}^{(i)}+\mathbf{B}_{\mathtt{eq}}\mathbf{x}^{(i)}+\mathbf{b}_{\mathtt{eq}}\right\|_1\right)$.

	Optimizer	Optimality Gap	Feasibility Gap	Time on CPU(msec)
Standard Solvers	Matpower [30]	0	0	290.00
	CVXOPT [31]	0	0	139.40
Neural Solvers	Projection-Based [21]	0.00194	0	221.24
	Penalty-Based [8]	0.01524	0.04819	0.16
	DC3 [24]	0.01032	0.04579	0.80
	Our	0.00203	0	0.76

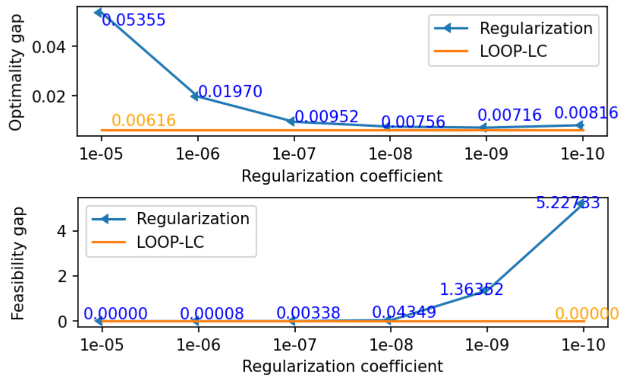

FIGURE 6. Optimality and feasibility results of image registration problems. Optimality gap is measured as $\frac{1}{N}\sum_{i=1}^{N}f(\mathbf{u}_{\mathrm{r}}^{(i)})$, while the feasibility gap is calculated as $\frac{1}{N}$ $(\|\max(\mathbf{A}_{\mathrm{r}}\mathbf{u}_{\mathrm{r}}-\epsilon\mathbf{I}_d,\mathbf{0})\|_1+\|\max(-\mathbf{A}_{\mathrm{r}}\mathbf{u}_{\mathrm{r}}-\epsilon\mathbf{I}_d,\mathbf{0})\|_1)$.

target image I_{t}. Moreover, let $\mathbf{f}_{\mathrm{r}}$ be the registration field that maps coordinates of I_{s} to coordinates of I_{t}. Given these definitions, the optimization problem can be written as, $\arg\min_{\mathbf{f}_{\mathrm{r}}}\int_{\mathcal{X}}\|I_s(\mathbf{f}_{\mathrm{r}}(x))-I_t(x)\|^2dx$. Often, $\mathbf{f}_{\mathrm{r}}$ is characterized by a displacement vector field $\mathbf{u}_{\mathrm{r}}$. This vector specifies the vector offset for each voxel: $\mathbf{f}_{\mathrm{r}}=\mathbf{I}_d+\mathbf{u}_{\mathrm{r}}$, where $\mathbf{I}_d$ is the identity transform. Hence, the problem transforms to,

$$\min \mathrm{f}(\mathbf{u}_{\mathrm{r}})=\int_{\mathcal{X}}\|I_s(x+\mathbf{u}_{\mathrm{r}}(x))-I_t(x)\|^2dx \tag{14}$$

Problem (14) is highly non-convex and in many applications, e.g., medical image analysis, the displacement $\mathbf{u}_{\mathrm{r}}$ must satisfy some regularization/smoothness conditions. Extensive prior works have devised various penalty terms to enforce the smoothness of the displacement fields. One such approach is to penalize the gradients' norms of the displacement along the x-axis and y-axis. In this paper, we enforce upper and lower bounds for gradients of $\mathbf{u}_{\mathrm{r}}$, thus, the feasible range of problem (14) can be defined as,

$$\mathcal{S}=\{-\epsilon\mathbf{I}_d\leq\mathbf{A}_{\mathrm{r}}\mathbf{u}_{\mathrm{r}}\leq\epsilon\mathbf{I}_d\} \tag{15}$$

where $\mathbf{A}_{\mathrm{r}}$ denotes the gradient operator along x&y axis. We choose $\epsilon=0.01$ in the paper.

1) DATASET

We use 25000 pairs of images from the MNIST dataset [33] for training.

2) COMPARISON

We compare $\mathcal{LOOP}-\mathcal{LC}$ against the regularization method [34] that utilizes mean squared error as the penalty term.

3) PARAMETERS

We use a customized Residual Neural Network (ResNet) [36] as the neural network architecture. The Tanh activation is added to the output layer for $\mathcal{LOOP}-\mathcal{LC}$ model. The learning rate is set to 0.001.

4) INTERIOR POINT FINDER

$\mathcal{S}^{\mathrm{Ref}}$ is fixed in the image registration problem. Therefore, we use $\mathbf{0}$ as the interior point in $\mathcal{LOOP}-\mathcal{LC}$.

5) RESULTS

The average per-instance time (using GPU) for the regularization method is $1.5576*10^{-5}$ seconds while

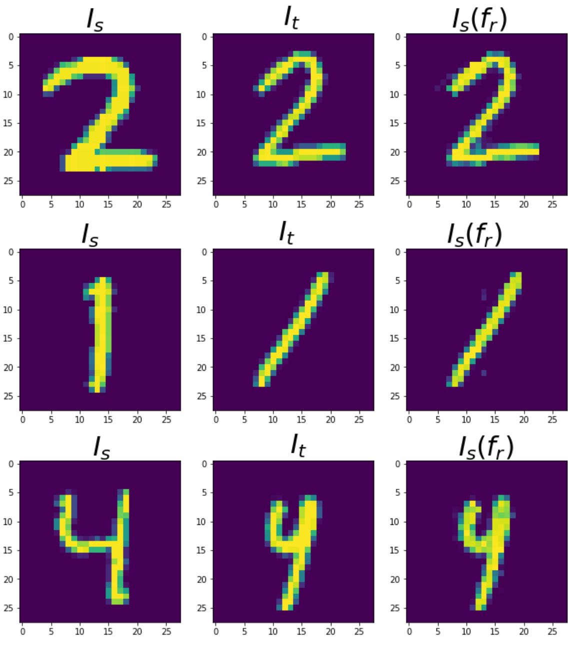

FIGURE 7. Examples of learning results of image registration problems. The source image and target image are in columns 1-2, and the results using $\mathcal{LOOP}-\mathcal{LC}$ model in column 3. A well-tuned registration function will produce $I_s(\mathbf{f}_{\mathrm{r}})$(column 3) similar to images in column 2. Our models perform well in various images while maintaining smooth displacements.

$\mathcal{LOOP}-\mathcal{LC}$ outputs the results in $1.7699 * 10^{-5}$ seconds. Optimality and feasibility results are shown in Figure 6. Although the regularization method slightly outperforms the $\mathcal{LOOP}-\mathcal{LC}$ in terms of speed, it yields a considerable feasibility gap. The $\mathcal{LOOP}-\mathcal{LC}$ model, free of parameter tuning, achieves high-quality (close-to-optimal) solutions while guaranteeing feasibility with respect to hard constraints. Figure 7 illustrates training results using $\mathcal{LOOP}-\mathcal{LC}$ model.

V. CONCLUSION

This paper introduces the $\mathcal{LOOP}-\mathcal{LC}$ model for solving an optimization problem with hard linear constraints. At its core, our method is a neural approximator that maps the inputs to an optimization problem with hard linear constraints to a high-quality feasible solution (near optimal). In a nutshell, our proposed model learns a neural approximator that maps the optimization's inputs to an optimal point in the ℓ_∞-norm unit ball and then maps the ℓ_∞-norm unit ball to the feasible set of the original problem through a gauge map. Unlike current learning-assisted solutions, our method is free of parameter tuning and removes iterations altogether. Our results on convex and non-convex optimization tasks showcase that the $\mathcal{LOOP}-\mathcal{LC}$ achieves close-to-optimal feasible solutions (with respect to hard constraints) while outperforming existing solutions in terms of solution time. Our proposed method is especially applicable to complex optimization problems with linear constraints where the interior points could be efficiently produced.

REFERENCES

[1] X.-H. Yu, G.-A. Chen, and S.-X. Cheng, "Dynamic learning rate optimization of the backpropagation algorithm," *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 669–677, May 1995.

[2] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1—Learning rate, batch size, momentum, and weight decay," 2018, *arXiv:1803.09820*.

[3] S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends® Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.

[4] J. Yang, T. Chen, M. Zhu, F. He, D. Tao, Y. Liang, and Z. Wang, "Learning to generalize provably in learning to optimize," 2023, *arXiv:2302.11085*.

[5] B. Amos, "Tutorial on amortized optimization," 2022, *arXiv:2202.00665*.

[6] B. Amos, S. Cohen, G. Luise, and I. Redko, "Meta optimal transport," 2022, *arXiv:2206.05262*.

[7] X. Yuan, S. Hu, W. Ni, R. P. Liu, and X. Wang, "Joint user, channel, modulation-coding selection, and RIS configuration for jamming resistance in multiuser OFDMA systems," *IEEE Trans. Commun.*, vol. 71, no. 3, pp. 1631–1645, Mar. 2023.

[8] X. Liu, Y. Lu, A. Abbasi, M. Li, J. Mohammadi, and S. Kolouri, "Teaching networks to solve optimization problems," 2022, *arXiv:2202.04104*.

[9] X. Pan, M. Chen, T. Zhao, and S. H. Low, "DeepOPF: A feasibility-optimized deep neural network approach for AC optimal power flow problems," 2020, *arXiv:2007.01002*.

[10] S. Liu, C. Wu, and H. Zhu, "Topology-aware graph neural networks for learning feasible and adaptive AC-OPF solutions," 2022, *arXiv:2205.10129*.

[11] M. Gao, J. Yu, Z. Yang, and J. Zhao, "A physics-guided graph convolution neural network for optimal power flow," *IEEE Trans. Power Syst.*, early access, Jan. 20, 2023, doi: 10.1109/TPWRS.2023.3238377.

[12] B. Chen, P. L. Donti, K. Baker, J. Z. Kolter, and M. Bergés, "Enforcing policy feasibility constraints through differentiable projection for energy optimization," in *Proc. 12th ACM Int. Conf. Future Energy Syst.*, New York, NY, USA, Jun. 2021, pp. 199–210.

[13] F. Detassis, M. Lombardi, and M. Milano, "Teaching the old dog new tricks: Supervised learning with constraints," in *Proc. NeHuAI@ ECAI*, 2020, pp. 44–51.

[14] 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, Mar. 2021.

[15] L. Zhang, Y. Chen, and B. Zhang, "A convex neural network solver for DCOPF with generalization guarantees," *IEEE Trans. Control Netw. Syst.*, vol. 9, no. 2, pp. 719–730, Jun. 2022.

[16] R. Nellikkath and S. Chatzivasileiadis, "Physics-informed neural networks for AC optimal power flow," *Electr. Power Syst. Res.*, vol. 212, Nov. 2022, Art. no. 108412.

[17] Y. Chen, L. Zhang, and B. Zhang, "Learning to solve DCOPF: A duality approach," *Electr. Power Syst. Res.*, vol. 213, Dec. 2022, Art. no. 108595.

[18] F. Fioretto, T. W. Mak, and P. V. Hentenryck, "Predicting AC optimal power flows: Combining deep learning and Lagrangian dual methods," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, pp. 1–12.

[19] S. Park and P. Van Hentenryck, "Self-supervised primal-dual learning for constrained optimization," 2022, *arXiv:2208.09046*.

[20] C. Tran, F. Fioretto, and P. Van Hentenryck, "Differentially private and fair deep learning: A Lagrangian dual approach," 2020, *arXiv:2009.12562*.

[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 Int. Conf. Commun., Control, Comput. Technol. Smart Grids (SmartGridComm)*, Nov. 2020, pp. 1–6.

[22] T. Frerix, M. Nießner, and D. Cremers, "Homogeneous linear inequality constraints for neural network activations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 3229–3234.

[23] P. Hruby, T. Duff, A. Leykin, and T. Pajdla, "Learning to solve hard minimal problems," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 5522–5532.

[24] P. L. Donti, D. Rolnick, and J. Z. Kolter, "DC3: A learning method for optimization with hard constraints," 2021, *arXiv:2104.12225*.

[25] R. Christian, P. Harsha, G. Perakis, B. Quanz, and I. Spantidakis, "End-to-end learning for optimization via constraint-enforcing approximators," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 1–22.

[26] D. Tabas and B. Zhang, "Computationally efficient safe reinforcement learning for power systems," 2021, *arXiv:2110.10333*.

[27] F. Blanchini and S. Miani, *Set-Theoretic Methods in Control*, vol. 78. Cham, Switzerland: Springer, 2008.

[28] I. Adler, M. G. Resende, G. Veiga, and N. Karmarkar, "An implementation of Karmarkar's algorithm for linear programming," *Math. Program.*, vol. 44, no. 1, pp. 297–335, 1989.

[29] H. Ronellenfitsch, M. Timme, and D. Witthaut, "A dual method for computing power transfer distribution factors," *IEEE Trans. Power Syst.*, vol. 32, no. 2, pp. 1007–1015, Mar. 2017.

[30] R. D. Zimmerman, C. E. Murillo-Sanchez, 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, Feb. 2011.

[31] L. Vandenberghe. (2010). *The CVXOPT Linear and Quadratic Cone Program Solvers*. [Online]. Available: http://cvxopt.org/documentation/coneprog.pdf

[32] Z. Murez, S. Kolouri, D. Kriegman, R. Ramamoorthi, and K. Kim, "Image to image translation for domain adaptation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4500–4509.

[33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Mar. 1998.

[34] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2813–2821.

[35] M. Li, Y. Du, J. Mohammadi, C. Crozier, K. Baker, and S. Kar, "Numerical comparisons of linear power flow approximations: Optimality, feasibility, and computation time," in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Jul. 2022, pp. 1–5.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

MEIYI LI received the B.S. and M.S. degrees in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2017 and 2020, respectively, and the Ph.D. degree from Carnegie Mellon University, Pittsburgh, PA, USA, in 2021. She is currently pursuing the Ph.D. degree with The University of Texas at Austin, Austin, TX, USA.

Her major field of study is electrical engineering, with a focus on power and energy systems optimization. Her research interests include encompass learning to optimize and distributed optimization within the power and energy domain. She received numerous accolades, including the prestigious ''Best of the Best'' Paper Award from the 2019 IEEE Power and Energy Society General Meeting. Her impressive academic record includes the National Scholarship for Outstanding Academic Achievements, in 2018, the Carnegie Institute of Technology Dean's Fellowship, in 2020, and the Distinguished Alumni Endowed Graduate Fellowship from The University of Texas at Austin, in 2022.

SOHEIL KOLOURI (Senior Member, IEEE) received the Ph.D. degree from the Biomedical Engineering Department, Carnegie Mellon University (CMU), in 2015.

He is currently an Assistant Professor with the Department of Computer Science, Vanderbilt University, Nashville, TN, USA, where he leads the Machine Intelligence and Neural Technologies (MINT) Laboratory. Prior to joining Vanderbilt University, he was a Research Scientist with HRL Laboratories, LLC., Malibu, CA, USA, where he led multiple DARPA programs on AI in autonomy. His research centered on pattern recognition in medical images. His research interests include applied mathematics, machine learning, and computer vision, with a specific focus on computational optimal transport and geometry. He is an Active Member of the Association for Computing Machinery (ACM). He was a recipient of the Bertucci Fellowship Award for outstanding graduate students from the College of Engineering, in 2014, and the Outstanding Dissertation Award from the Biomedical Engineering Department, in 2015. His work has been recognized by numerous international awards, including the Best Paper Award from the 2022 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).

JAVAD MOHAMMADI (Senior Member, IEEE) received the Ph.D. degree from the Electrical and Computer Engineering Department, Carnegie Mellon University (CMU), in 2016.

He is currently an Assistant Professor with the Department of Civil, Architectural, and Environmental Engineering, The University of Texas at Austin (UT Austin). Before joining UT Austin, he was a Faculty Member with the Electrical and Computer Engineering Department, CMU. AFOSR, ARPA-E, and the Department of Energy support his grid modernization efforts. His research on building energy management has received financial support from the local governments and institutions, such as the Sloan Foundation. His research interests include distributed decision-making in networked cyber-physical systems, including energy networks and electrified transportation systems. As a graduate student, he was a recipient of the Innovation Fellowship from the Swartz Center for Entrepreneurship.