Neural Networks for AC Optimal Power Flow: Improving Worst-Case Guarantees during Training

Bastien Giraud*, Rahul Nellikath*, Johanna Vorwerk*, Maad Alowaifeer†, Spyros Chatzivasileiadis*

*Department of Wind and Energy Systems, Technical University of Denmark, Kongens Lyngby, Denmark
†Electrical Engineering Department, SDAIA-KFUPM Joint Research Center for Artificial Intelligence
King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
bagir@dtu.dk, alowaifeer@kfupm.edu.sa

Abstract—The AC Optimal Power Flow (AC-OPF) problem is central to power system operation but challenging to solve efficiently due to its nonconvex and nonlinear nature. Neural networks (NNs) offer fast surrogates, yet their black-box behavior raises concerns about constraint violations that can compromise safety. We propose a verification-informed NN framework that incorporates worst-case constraint violations directly into training, producing models that are both accurate and provably safer. Through post-hoc verification, we achieve substantial reductions in worst-case violations and, for the first time, verify all operational constraints of large-scale AC-OPF proxies. Practical feasibility is further enhanced via restoration and warm-start strategies for infeasible operating points. Experiments on systems ranging from 57 to 793 buses demonstrate scalability, speed, and reliability, bridging the gap between ML acceleration and safe, real-time deployment of AC-OPF solutions — and paving the way toward data-driven optimal control.

Index Terms—AC Optimal Power Flow, Neural Network Verification, Trustworthy Machine Learning

I. INTRODUCTION

The AC Optimal Power Flow (AC-OPF) problem is fundamental to power system operation, control, and electricity market operation. It determines the optimal steady-state operating point of the grid while satisfying physical and operational constraints. Increasing uncertainty from renewable generation, the proliferation of distributed energy resources, and the complexity of emerging market mechanisms necessitate solving this problem across a wide range of operating conditions. In its exact formulation, AC-OPF is a nonconvex and nonlinear optimization problem [1], which makes computing globally optimal solutions particularly challenging.

Recent years have witnessed a surge of interest in leveraging machine learning (ML) to accelerate and enhance power system decision-making. ML-based surrogates can rapidly evaluate thousands of operating points in the time of a single traditional optimization, enabling scalable uncertainty quantification and supporting real-time predictive optimal control with AC-OPF solutions. Neural networks (NNs), in particular, have emerged as promising candidates due to their ability to approximate highly nonlinear mappings with remarkable accuracy. A fundamental barrier to the deployment of NNs in

safety-critical applications, such as power systems, lies in their black-box nature. Although NNs can produce highly accurate approximations, ensuring that their outputs satisfy all physical and operational constraints across the continuous input space remains a major challenge. This issue is particularly acute for AC-OPF, where violations of voltage, line flow, or generator limits can result in unsafe or unstable operating conditions. Therefore, any ML-based solution must not only accurately approximate the OPF mapping but also provide rigorous guarantees of constraint satisfaction.

To address this challenge, a range of approaches has been proposed. Physics-informed NNs incorporate domain knowledge directly into AC-OPF models [2]. Other strategies focus on accelerating or constraining the learning process, such as learning warm-starts for traditional solvers [3], [4], embedding differentiable optimization layers to enforce hard constraints during training [5], or penalizing violations through dual-feasibility adjustments [6]. More recently, NN verification methods have attracted significant attention as a means to formally certify constraint satisfaction [7]–[10].

Despite this progress, important limitations remain. In prior work [2], [11], [12], we developed methods to rigorously quantify worst-case constraint violations of NNs for both DC-OPF and AC-OPF. Early approaches reformulated NNs as mixed-integer linear programs (MILPs) to cast verification as an optimization problem, providing valuable diagnostic insights but not enabling direct improvements during training [11]. Subsequent methods incorporated MILP-based verification into training to reduce worst-case violations, yet required multiple NNs and computationally expensive optimization at every step, leading to slowdowns and poor scalability for large networks [13].

In this paper, we advance this line of research by proposing a novel NN-based framework that approximates AC-OPF solutions while explicitly minimizing worst-case operational constraint violations during training. Unlike prior approaches, our method is verification-informed and integrates worst-case violations directly into the training loop, yielding models that are both accurate and provably safer. Through post-hoc verification, we demonstrate for the first time that all operational constraints of large-scale AC-OPF proxies can be formally verified, achieving significantly reduced worst-case violations compared to baseline NNs without compromising predictive performance. Furthermore, we address the inherent difficulty of ensuring feasibility across the full input domain by exploring complementary strategies, including feasibility restoration and warm-starting classical solvers, to guarantee physically realizable operating points. We objectively compare

two AC-OPF proxy formulations to evaluate their practicality for penalizing worst-case violations while preserving solution quality. Experiments on test systems ranging from 57 to 793 buses confirm that the proposed approach offers substantial computational gains over conventional OPF solvers with minimal accuracy loss. The contributions of this paper are twofold:

- We develop a verification-informed training framework for NNs that minimizes worst-case AC-OPF constraint violations, laying the groundwork for future real-time optimal control applications.
- For the first time, we verify all operational constraints of a large-scale AC-OPF proxy and demonstrate practical applicability by evaluating feasibility restoration and warm-start strategies for infeasible operating points on benchmark systems of up to 793 buses.

The remainder of the paper is organized as follows. Section II presents the AC-OPF formulation and the NN training workflow. Section III describes the verification procedure, its integration into training, and the recovery of physically realizable operating points. In Section IV, we present both statistical and formally verified constraint violations and demonstrate how feasibility restoration ensures physically realizable NN predictions. Finally, Section V discusses the limitations of the proposed approach, outlines future research directions, and concludes the paper.

II. LEARNING AC OPTIMAL POWER FLOW

In this section, we first formally define the AC-OPF problem. We then describe two NN architectures designed to approximate its solution efficiently, highlighting how each incorporates physical knowledge and operational constraints.

A. Optimal Power Flow Problem

The AC-OPF seeks the most economical operating point of a power system while satisfying physical and operational constraints. We consider a system with $\mathcal{N}_g$ generators, $\mathcal{N}_b$ buses, $\mathcal{N}_d$ loads, and $\mathcal{N}_l$ transmission lines. The standard objective is to minimize the total cost of active power generation:

$$\min_{\mathbf{P}_g, \mathbf{v}} \quad \mathbf{c}_p^{\mathsf{T}} \mathbf{P}_g, \tag{1}$$

where $\mathbf{P}_g$ is the active power generator setpoints, $\mathbf{v}$ is the vector of complex bus voltages, and $\mathbf{c}_p$ collects the generation cost coefficients. At each bus $n \in \mathcal{N}_b$, power balance requires

$$p_n = p_n^g - p_n^d, \quad q_n = q_n^g - q_n^d, \tag{2}$$

where p_n and q_n are the bus injections, p_n^g and q_n^g denote the total active and reactive generation, and p_n^d and q_n^d the corresponding demand. The nonlinear AC power flow equations relate bus voltages to active and reactive power injections. For compactness, we define $\mathbf{v} = [(\mathbf{v}^r)^{\mathsf{T}}, (\mathbf{v}^i)^{\mathsf{T}}]^{\mathsf{T}} \in \mathbb{R}^{2N_b}$, where $\mathbf{v}^r$ and $\mathbf{v}^i$ collect the real and imaginary voltage components. Following [14], the AC power injections at bus n can be expressed compactly as

$$p_n = \mathbf{v}^{\mathsf{H}} H_{p_n} \mathbf{v}, \quad q_n = \mathbf{v}^{\mathsf{H}} H_{q_n} \mathbf{v}, \tag{3}$$

where H_{P_n} and H_{Q_n} are Hermitian matrices capturing the network topology and branch admittances. This provides a compact representation of the nonlinear power flow equations.

Generator limits impose bounds on feasible injections:

$$\underline{p}_n^g \leq \mathbf{v}^{\mathsf{H}} H_{p_n} \mathbf{v} + p_n^d \leq \overline{p}_n^g,$$
$$\underline{q}_n^g \leq \mathbf{v}^{\mathsf{H}} H_{q_n} \mathbf{v} + q_n^d \leq \overline{q}_n^g, \tag{4}$$

for all $n \in \mathcal{N}_g$. Bus voltage magnitudes and branch flows are constrained as

$$\underline{V}_n \leq \mathbf{v}^{\mathsf{H}} H_{V_n} \mathbf{v} \leq \overline{V}_n, \quad \forall n \in \mathcal{N}_b,$$
$$\mathbf{v}^{\mathsf{H}} H_{I_{mn}} \mathbf{v} \leq \overline{l}_{mn}, \quad \forall (m,n) \in \mathcal{N}_l, \tag{5}$$

Finally, the slack bus N_{sb} fixes the voltage angle reference via $v_{N_{sb}}^i = 0$. Taken together, the objective Equation (1), the balance relations Equation (2), and the constraints Equations (3) to (5) constitute the full AC-OPF formulation.

B. Neural Networks for Optimal Power Flow

Given the highly nonlinear and nonconvex nature of AC-OPF, we next describe how NNs can be trained to efficiently approximate its solution. A NN is a parametric function $f_\theta : \mathbb{R}^n \to \mathbb{R}^m$ with parameters θ (weights $\mathbf{w}$ and biases $\mathbf{b}$), trained to approximate a mapping from inputs x_i to targets y_i. In the context of AC-OPF, the NN typically takes the system load profile as input and outputs approximate control variables that resemble the optimal solution. With sufficient size, NNs can closely approximate the mapping from power system demands to OPF setpoints, enabling rapid evaluation of multiple operating points.

1) Power Neural Network: The first NN architecture we implement maps the load demand directly to generator setpoints and voltages, $f_\theta(\mathbf{p}_d, \mathbf{q}_d) \longrightarrow (\mathbf{p}_g, \mathbf{V}_g)$, where $\mathbf{p}_g, \mathbf{V}_g \in \mathbb{R}^{\mathcal{N}_g}$ denote the active power and voltage magnitude setpoints of all generators. This approach is intuitive, as these variables correspond directly to the generators' controllable inputs. The NN is trained using a supervised loss term combined with penalty terms for constraint violations. The supervised part is the mean squared error (MSE) between the predicted outputs and the labeled training data:

$$\mathcal{L}_{\mathrm{MSE}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(f_\theta(x_i) - y_i \right)^2. \tag{6}$$

Constraint violations are penalized using

$$\mathcal{L}_z = \sum_{n \in \mathcal{N}} \left[\sigma(z_n - \overline{z}_n)^2 + \sigma(\underline{z}_n - z_n)^2 \right], \tag{7}$$

where $\sigma(\cdot) = \max(0, \cdot)$ denotes the rectified linear unit (ReLU) activation, and z_n is the variable of interest over the index set $\mathcal{N}$. Specifically, we apply Equation (7) to all generator active powers p_n^g and voltage magnitudes V_n for $n \in \mathcal{N}_g$, yielding $\mathcal{L}_{P_g}$ and $\mathcal{L}_V$, respectively. The overall loss is then:

$$\mathcal{L}_{\mathrm{power}} = \Lambda_{\mathrm{MSE}} \mathcal{L}_{\mathrm{MSE}} + \Lambda_{P_g} \mathcal{L}_{P_g} + \Lambda_{V_g} \mathcal{L}_{V_g}, \tag{8}$$

where $\Lambda_{\mathrm{MSE}}, \Lambda_{P_g}$, and Λ_{V_g} are hyperparameters weighting the contribution of each term.

While predicting $(\mathbf{p}_g, \mathbf{V}_g)$ is straightforward, recovering the full system state requires solving a power flow, which adds computational overhead. Moreover, conventional NN verification methods cannot be applied to these reconstructed states, since the iterative nature of the power flow solver cannot be represented in a differentiable computational graph. These

challenges motivate alternative NN architectures that predict richer sets of variables, enabling the integration of physical knowledge and feasibility constraints directly into the learning process.

2) Voltage Neural Network: The second NN architecture predicts the rectangular bus voltages directly: $f_\theta(\mathbf{p}_d, \mathbf{q}_d) \longrightarrow (\mathbf{v}^r, \mathbf{v}^i)$, where $\mathbf{v}^r, \mathbf{v}^i \in \mathbb{R}^{\mathcal{N}_b}$ are the real and imaginary voltage components at all buses. By predicting the full system state, this approach eliminates the need for additional power flow computations and allows direct penalization of constraint violations during training. It also improves verification efficiency, since each operational constraint depends only on a subset of voltages. This means we only need to verify over a smaller portion of the NN, reducing the number of ReLUs and overall nonlinearity. For example, if we check a branch flow constraint between nodes k and m, we only need the outputs $\mathbf{v}_k^r, \mathbf{v}_k^i$ and $\mathbf{v}_m^r, \mathbf{v}_m^i$, and can ignore all other voltage outputs.

The bus voltage magnitude is computed as $V_n = \sqrt{(v_n^r)^2 + (v_n^i)^2}$, $\forall n \in \mathcal{N}_b$. The bus current injections follow from the bus admittance matrix, $\mathbf{Y}_{\text{bus}} \in \mathbb{C}^{N_b \times N_b}$, which encodes the network topology and branch admittance parameters. To operate on the stacked real and imaginary voltage components, $\mathbf{v} = [(\mathbf{v}^r)^\mathsf{T}, (\mathbf{v}^i)^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{2N_b}$, we define the rectangular admittance matrix

$$\mathbf{Y}_{\text{bus}}^{\text{rect}} = \begin{bmatrix} \text{Re}(\mathbf{Y}_{\text{bus}}) & -\text{Im}(\mathbf{Y}_{\text{bus}}) \\ \text{Im}(\mathbf{Y}_{\text{bus}}) & \text{Re}(\mathbf{Y}_{\text{bus}}) \end{bmatrix} \in \mathbb{R}^{2N_b \times 2N_b}. \quad (9)$$

The bus currents are then collected in the vector $\mathbf{i} = [(\mathbf{i}^r)^\mathsf{T}, (\mathbf{i}^i)^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{2N_b}$, where $\mathbf{i}^r$ and $\mathbf{i}^i$ denote the real and imaginary components, respectively, and satisfy

$$\mathbf{i} = \mathbf{Y}_{\text{bus}}^{\text{rect}} \mathbf{v}. \quad (10)$$

Branch currents are computed analogously using branch admittance matrices, which capture the contributions at the "from" and "to" ends of each branch. In rectangular coordinates, the branch admittances are

$$\mathbf{Y}_f^{\text{rect}} = \begin{bmatrix} \mathbf{G}_f & -\mathbf{B}_f \\ \mathbf{B}_f & \mathbf{G}_f \end{bmatrix}, \quad \mathbf{Y}_t^{\text{rect}} = \begin{bmatrix} \mathbf{G}_t & -\mathbf{B}_t \\ \mathbf{B}_t & \mathbf{G}_t \end{bmatrix} \in \mathbb{R}^{2N_l \times 2N_b}, \quad (11)$$

which are stacked to form the full branch-current matrix

$$\mathbf{Y}_l^{\text{rect}} = \begin{bmatrix} \mathbf{Y}_f^{\text{rect}} \\ \mathbf{Y}_t^{\text{rect}} \end{bmatrix} \in \mathbb{R}^{4N_l \times 2N_b}. \quad (12)$$

The branch currents are collected in the vector

$$\mathbf{i}_l = [(\mathbf{i}_f^r)^\mathsf{T}, (\mathbf{i}_f^i)^\mathsf{T}, (\mathbf{i}_t^r)^\mathsf{T}, (\mathbf{i}_t^i)^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{4N_l}, \quad \mathbf{i}_l = \mathbf{Y}_l^{\text{rect}} \mathbf{v}, \quad (13)$$

and the corresponding branch current magnitudes for each branch $(m,n) \in N_l$ are $l_{mn} = \sqrt{(i_{mn}^r)^2 + (i_{mn}^i)^2}$.

Finally, the bus power injections can be expressed in terms of real and imaginary voltage and current components:

$$\mathbf{p}_n = \mathbf{v}^r \odot \mathbf{i}^r + \mathbf{v}^i \odot \mathbf{i}^i, \quad \forall n \in N_b, \quad (14)$$

$$\mathbf{q}_n = \mathbf{v}^i \odot \mathbf{i}^r - \mathbf{v}^r \odot \mathbf{i}^i, \quad \forall n \in N_b, \quad (15)$$

from which generator setpoints can be recovered using the nodal balance equations (Equation (2)).

The branch currents and bus injections defined above provide a direct way to evaluate operational constraints within the NN training loss. Notably, the branch currents $\mathbf{i}_l$ depend linearly on the bus voltages through the branch admittance

matrix $\mathbf{Y}_l^{\text{rect}}$, the voltage and current magnitudes are obtained via elementwise norms, and the bus power injections are bilinear in voltage and current.

The reactive power penalty is obtained applying Equation (7) to all generator reactive powers q_n^g yielding $\mathcal{L}_{Q_g}$. The branch flow penalty is

$$\mathcal{L}_l = \sum_{mn \in N_l} \sigma(|l_{mn}| - \bar{l}_{mn})^2, \quad (16)$$

and the nodal balance penalty is

$$\mathcal{L}_{\text{bal}} = \sum_{n \in \mathcal{N}_b} \left[(i_n^r - \sum_{m \in \mathcal{N}_n} i_{nm}^r)^2 + (i_n^i - \sum_{m \in \mathcal{N}_n} i_{nm}^i)^2 \right], \quad (17)$$

The complete loss function combines the MSE loss with penalties for generation limits, voltage bounds, branch flows, and nodal balance:

$$\mathcal{L}_{\text{voltage}} = \Lambda_{\text{MSE}} \mathcal{L}_{\text{MSE}} + \Lambda_{P_g} \mathcal{L}_{P_g} + \Lambda_{Q_g} \mathcal{L}_{Q_g} \\ + \Lambda_{V_m} \mathcal{L}_{V_m} + \Lambda_l \mathcal{L}_l + \Lambda_{\text{bal}} \mathcal{L}_{\text{bal}}, \quad (18)$$

where each term is weighted by a hyperparameter Λ.

III. NEURAL NETWORK VERIFICATION

This section defines the NN verification problem and shows how it rigorously evaluates constraint satisfaction. We then describe methods to integrate worst-case guarantees during training, enabling the network to minimize violations proactively. Finally, we outline strategies to recover feasible solutions when violations persist ensuring safe operating points.

A. Worst-Case Guarantees for Neural Networks

While incorporating operational constraints into the NN training loss encourages feasible setpoints, it does not provide formal guarantees. Even small approximation errors can lead to violations of voltage, current, or generator limits. NN verification addresses this limitation by rigorously certifying that a trained NN satisfies all feasibility constraints across its input domain. Specifically, for all inputs $x \in \mathcal{X}$, we seek to ensure $f_\theta(x) \in \mathcal{F}$, $\forall x \in \mathcal{X}$, where $\mathcal{F}$ denotes the feasible set. In power systems, such guarantees can translate into certified bounds on constraint violations, ensuring that the NN produces safe and reliable setpoints under extreme or adversarial conditions. Early verification attempts cast the NN as a MILP [11], formulating an optimization problem to identify the worst-case violation ν of operational constraints over the input domain:

$$\max_{x \in \mathcal{X}} \nu \quad \text{s.t.} \quad f_\theta(x), \quad (19)$$

where, for example, violations ν_{P_g}, ν_{Q_g} and ν_{V_m} can be defined using:

$$\nu_{z_n} = \max_{x \in \mathcal{X}}(\sigma(z_n - \bar{z}_n), \sigma(\underline{z}_n - z_n), 0), \quad (20)$$

and similarly for line flows and current balance:

$$\nu_l = \max_{x \in \mathcal{X}} \left(|\hat{l}_{mn}|/\bar{l}_{mn} - 1, 0 \right),$$

$$\nu_{bal} = \max_{x \in \mathcal{X}} \left[(\hat{i}_n^r - \sum_{m \in \mathcal{N}_n} \hat{i}_{nm}^r)^2 + (\hat{i}_n^i - \sum_{m \in \mathcal{N}_n} \hat{i}_{nm}^i)^2 \right]. \quad (21)$$

While this MILP-based formulation provides exact guarantees, it scales poorly; large NNs require many binary variables, and incorporating nonlinearities can lead to mixed-integer
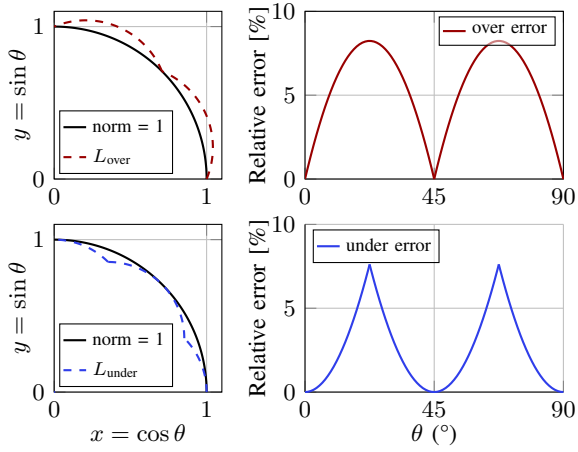
Fig. 1: Over- and under-approximating a unit phasor's magnitude using the α–max and β–min formulas (24) and (26).

quadratic or quadratically constrained programs (MIQCQP) [2], this method is intractable for realistic power system models. As a result, these methods are typically limited to *post-hoc verification*, performed only after training.

Recent advances in verification tools, such as α-β-CROWN [7]–[9], [15], together with community efforts like the VNN-Comp competition [10], have greatly improved scalability and speed. This makes it feasible to integrate verification of linear and piecewise-linear operations directly into the computational graph, enabling efficient penalization of worst-case constraint violations during training. α-β-CROWN uses nonlinear relaxations and branch-and-bound to produce tight bounds for nonlinear quantities, but is computationally intensive. In contrast, α-CROWN handles only linear quantities, yet produces bounds quickly, making it well-suited to use during NN training.

B. Minimizing Worst-Case Violations During Training

Ensuring that a NN respects worst-case operational constraints is critical for safe AC-OPF predictions. In this work, we leverage the linear bound propagation method called α-CROWN to integrate worst-case violation minimization directly into NN training. Worst-case constraint violations are penalized efficiently at each epoch by propagating linear upper and lower bounds on operational constraints through the network, which are aggregated in a worst-case loss term:

$$\mathcal{L}_{\mathrm{wc}} = \Lambda_{wc}(\nu_{P_g} + \nu_{Q_g} + \nu_{V_m} + \nu_l + \nu_{\mathrm{bal}}). \quad (22)$$

While the constraint violations penalized in (8) and (18) depend on the training data, the worst-case violations are defined over the entire input domain of the dataset, making them independent of the training dataset's quality.

To make this bound propagation tractable during training, the nonlinearities in the AC-OPF problem must be relaxed, in particular: (i) magnitudes of complex variables such as bus voltages and branch currents, and (ii) bilinear products of voltages and currents appearing in the active and reactive power injections. In the following, we describe the building blocks used to handle these nonlinearities and enable tractable worst-case bound computation with α-CROWN.

1) α-max β-min: Many power system quantities, such as bus voltages and branch currents, involve phasor magnitudes. For $x = x^r + jx^i \in \mathbb{C}$, the magnitude is $|x| =$ $\sqrt{(x^r)^2 + (x^i)^2}$, a nonlinear operation not directly compatible with linear bound-propagation methods. To obtain tractable guarantees, we approximate the Euclidean norm using the α–max β–min formula:

$$|x| \approx \alpha \max(|x^r|, |x^i|) + \beta \min(|x^r|, |x^i|), \quad (23)$$

where $\alpha, \beta \geq 0$. With suitable parameters, this yields guaranteed over- or under-approximations. Because the formulation only involves $\max$, $\min$, and absolute values, it can be encoded with linear constraints, making it compatible with both MILP and linear bound-propagation frameworks like α-CROWN.

For over-approximation, we use

$$L_{\mathrm{over}} = \max(|x^r|, |x^i|) + (\sqrt{2} - 1) \min(|x^r|, |x^i|), \quad (24)$$

which guarantees $L_{\mathrm{over}} \geq |x|$ for all $\theta \in [0°, 90°]$ with a worst-case error of about 8% and an average error of about 5%, requiring only 6 additional binaries.

For under-approximation, two candidates are defined:

$$L_\infty = \max(|x^r|, |x^i|), \qquad L_1 = \frac{1}{\sqrt{2}}(|x^r| + |x^i|). \quad (25)$$

The bound

$$L_{\mathrm{under}} = \max(L_\infty, L_1) \quad (26)$$

satisfies $L_{\mathrm{under}} \leq |x|$ for all $\theta \in [0°, 90°]$ with a worst-case error of about 8% and an average error of about 3%, requiring 8 additional binaries. See Figure 1 for illustrative plots for the α-max β-min approximation method.

2) McCormick Relaxations: The active and reactive power injections at the buses are bilinear in the real and imaginary parts of voltages and currents, i.e., (v_r, v_i) and (i_r, i_i). To enable α-CROWN to compute bounds on these quantities, we replace bilinear products with linear relaxations. A common approach is to use *McCormick relaxations*, which describe the convex hull of a bilinear term given variable bounds. For $z = xy$ with $x \in [x^L, x^U]$, $y \in [y^L, y^U]$, the convex hull is defined by the following inequalities:

$$z \geq x^L y + y^L x - x^L y^L, \quad z \geq x^U y + y^U x - x^U y^U,$$
$$z \leq x^U y + y^L x - x^U y^L, \quad z \leq x^L y + y^U x - x^L y^U. \quad (27)$$

Each bilinear product (e.g., $v_r i_r$, $v_i i_i$, $v_i i_r$, $v_r i_i$) is relaxed using (27), resulting in a linear outer-approximation of the nonlinear injections. Importantly, in our setting, these relaxations are not enforced as explicit constraints. Instead, they are used to compute upper and lower bounds on bilinear terms, enclosing the true nonlinear injection.

The bounds on v and i are provided by α-CROWN during training and are iteratively refined as the worst-case violations decrease. This iterative tightening improves the quality of the McCormick relaxation, leading to progressively tighter bounds on the power injections.

C. Obtaining a Physically Realizable State

Ideally, NN predictions could be deployed directly, as straight inference is dramatically faster than solving a full AC-OPF. However, even with worst-case violations minimized during training, some outputs may slightly violate the nonconvex AC-OPF constraints. In such cases, corrective strategies such as feasibility restoration or warm-starting are required to guarantee a physically realizable solution.

a) Feasibility Restoration: Consider a NN prediction $\hat{\mathbf{x}}$ that is infeasible with respect to the nonconvex AC-OPF

TABLE I: Test System and Neural Network Training Parameters.

PGLib Network	N_b	N_g	N_d	Nominal load [MW]	NN parameters hidden / batch / lr
57-bus	57	7	42	1 250	25 / 25 / 5e-4
118-bus	118	54	99	4 242	50 / 50 / 10e-4
300-bus	300	69	193	23 847	75 / 75 / 10e-4
793-bus	793	88	503	13 218	100 / 100 / 20e-4

constraints. A feasible dispatch $\mathbf{x}^\star$ can be recovered by solving the following least-squares optimization problem:

$$\min_{\mathbf{x}} \quad \sum_{k \in \mathcal{X}} \left(x_k - \hat{x}_k\right)^2 \tag{28a}$$

$$\text{s.t.} \quad \mathbf{x} \in \mathcal{F}, \tag{28b}$$

where $\mathcal{F}$ denotes the feasible region defined by Equations (2) to (5), and $\mathcal{X}$ is the set of OPF variables. The solution $\mathbf{x}^\star$ yields a feasible point near the NN prediction, offering a practical method for feasibility restoration.

b) Warm Start: Alternatively, the NN prediction can initialize a conventional AC-OPF solver. If close to optimal, convergence is fast, reducing computation. If infeasible or far from optimal, the warm-start may offer little benefit or even slow convergence.

IV. CASE STUDIES

This section first outlines the case study settings, then evaluates NN performance on a held-out test set and through formal verification. Next, we show how worst-case violations can be systematically reduced, and finally, assess feasibility restoration and warm-start strategies for obtaining physically realizable solutions.

A. Case Study Settings

We evaluate our approach on benchmark systems from the PGLib-OPF library (v23.07) [16], with the system characteristics summarized in Table I. The 793-bus system was modified by aggregating multiple generators per bus into a single equivalent generator and converting the quadratic cost objective to a linear one. For each test case, we generate 11 000 demand scenarios by sampling loads between 60% and 100% of their nominal values. The sampling follows a Kumaraswamy(1.6, 2.8) distribution, with a Pearson correlation coefficient of 0.75 applied to capture realistic correlations between loads. Each dataset is split into 8 000 training samples, 2 000 validation samples, and 1 000 test samples.

For each test system, we train four NNs: two voltage NNs and two power NNs. The "base" networks are trained using the standard loss functions (8) and (18), while the "crown" networks incorporate worst-case violation penalties via (22) using α-CROWN. Note that for the power NNs, worst-case penalties can only be applied to active power and generator voltage magnitudes, as the method does not allow computing worst-case guarantees for other quantities. This setup allows us to evaluate the impact of worst-case guarantees on both accuracy and feasibility. The NNs are implemented in PyTorch [17] and trained with the ADAM optimizer [18]. All models have three hidden layers, with layer size, learning rate, and batch size adapted to each test system (see Table I). The loss weights are tuned via a 10-iteration Bayesian optimization sweep. All models undergo 1 000 training epochs, with structured pruning introduced after 500 epochs to zero out 50% of the weights and promote better generalization. The

AC-OPF, feasibility restoration and warm-start optimizations are implemented in PowerModels.jl [19] and JuMP [20], and accessed from Python through PandaModels.jl [21] and PyCall.jl [22]. IPOPT [23] is used as the underlying solver. To reduce the influence of outliers, we discard the 10 fastest and 10 slowest runs for each method. Neural network worst-case guarantees are obtained by using α-β-CROWN [7]–[9], [15]. All simulations are run on the DTU HPC cluster [24], and the code is publicly available on GitHub [25].

B. Statistical Violations

We assess the quality of the trained NNs by examining their generalization performance and the resulting constraint violations on the test sets of 1 000 unseen samples. The results are summarized in Table II. Overall, the networks generalize well, achieving low root mean squared error (RMSE) on their respective prediction targets.

Constraint violations are generally minor, indicating that the NNs produce near-feasible operating points. The most notable exceptions occur for the Power NN on the 118-bus system, which struggles to satisfy branch flow limits (see $\nu_l^{\max}$). For the larger 300- and 793-bus systems, the power flow did not converge (reported as N/A in Table II) when using the Power NN's predicted power and voltage setpoints, further underscoring the inherent challenge of achieving a fully feasible system state when predicting only generator setpoints.

C. Verified Worst-Case Violations

As the violations obtained on a test set are only a statistical measure, we rigorously verify the constraint violations over the entire input domain to analyze the NNs performance. Moreover, incorporating worst-case violations into training allows us to successfully reduce them.

1) Initial Input Space: To obtain the worst-case violations of the trained NN, we used the α-β-CROWN NN verification tool. The α-β-CROWN framework employs optimized nonlinear relaxation techniques, allowing us to pass nonlinear constraints directly to the tool [26]. It then approximates upper and lower bounds for these problems. However, since these tools are primarily designed for verifying robustness properties, they must be called iteratively to obtain the worst-case violations. Moreover, the resulting bounds are still over-approximations of the true problem, and, similar to MILP optimization, a branch-and-bound approach is used to achieve convergence. We plan to develop an α-β-CROWN-based framework for obtaining NN worst-case guarantees at a later stage.

The last 2 rows of Table II, and Table III, show the verified worst-case violations obtained using α-β-CROWN. From the results, we observe that the NNs trained with worst-case penalties (see 'crown' column) effectively reduce worst-case violations across all metrics by at least 50%. For the 57- and 118-bus systems, incorporating worst-case violations into the loss function completely eliminates voltage and line flow constraint violations across the entire dataset. The relatively large violations observed for P_g and Q_g in the voltage NNs occur because α-β-CROWN only run for 100 seconds in the case of 57 and 118 bus systems, and 300 seconds in the case of 300 and 793 bus systems, and in many cases, the relaxations remained loose and had not yet converged. Notably, after including the worst-case violations in the loss function, the worst-case real power violations are reduced to

TABLE II: Performance Averaged Over Test Set and Statistical[†] and Verified[°] Worst-Case Violations

	57-bus				118-bus				300-bus				793-bus			
	Power NN		Voltage NN		Power NN		Voltage NN		Power NN		Voltage NN		Power NN		Voltage NN	
	base	crown	base	crown	base	crown	base	crown	base	crown	base	crown	base	crown	base	crown
RMSE	0.04	0.05	0.01	0.01	0.04	0.04	0.01	0.02	0.26	0.27	0.08	0.10	0.03	0.03	0.10	0.08
$\nu_{P_g}^{max}$ [pu][†]	0.06	0.11	0.73	0.89	0.24	0.26	0.30	0.53	0.00	0.00	1.60	1.45	0.00	0.00	3.44	3.99
$\nu_{Q_g}^{max}$ [pu][†]	0.00	0.00	0.30	0.79	0.00	0.00	0.08	0.09	N/A	N/A	0.91	1.39	N/A	N/A	0.53	0.43
ν_l^{max} [pu][†]	0.00	0.00	0.00	0.00	8.24	8.24	0.00	0.00	N/A	N/A	0.02	0.38	N/A	N/A	0.03	0.05
$\nu_{V_m}^{max}$ [pu][†]	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.13	0.09	0.00	0.00	0.00	0.00
ν_{bal}^{max} [pu][†]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	N/A	N/A	1.4e-3	1.4e-3	N/A	N/A	0.00	0.00
$\nu_{P_g}^{max}$ [pu][°]	0.72	0.57	5.69	2.21	11.7	0.67	14.7	1.22	138	23.0	548	20.0	16.0	3.83	86.0	10.0
$\nu_{V_{m,g}}^{max}$ [pu][°]	0.00	0.00	0.09	0.00	0.00	0.00	0.94	0.00	0.55	0.17	1.12	0.48	0.70	0.05	1.51	1.24

roughly 10% of the nominal load. Although still significant, this represents a substantial improvement compared to the base NN, which exhibited worst-case generation constraint violations of approximately 200% of the nominal load.

2) Reducing the Input Space: We next investigate the relationship between input space boundaries and worst-case violations. To this end, we reduce the input domain $\{\mathbf{p}_d, \mathbf{q}_d\} \in \mathbf{D}$ using a parameter $\delta \in [0.0, 0.2]$ such that, $(0.6 + \delta)\,x_d^{max} \leq x_d \leq (1.0 - \delta)\,x_d^{max}$, for $x_d \in \{p_d, q_d\}$. We then compute the worst-case violations within these reduced domains for the crown NNs.

The results are presented in Figure 2 and Figure 3. In these figures, the y-axis shows the worst-case violation over the full input domain, along with the percentage of the violation remaining when the input domain is progressively reduced. We observe monotonically decreasing trends, indicating that shrinking the input space consistently reduces the magnitude of the worst-case violations. Unlike the observations in [11], we find that reducing the input domain does not necessarily drive all worst-case violations to zero. As discussed in Section IV-C1, the worst-case violation problem did not converge to optimality due to its extremely nonlinear nature. Consequently, the results shown represent upper bounds of the true violations, which remain loose even for very small input domains after linear relaxation. Nevertheless, it is important to note that even for small input domains, some violations may still persist.
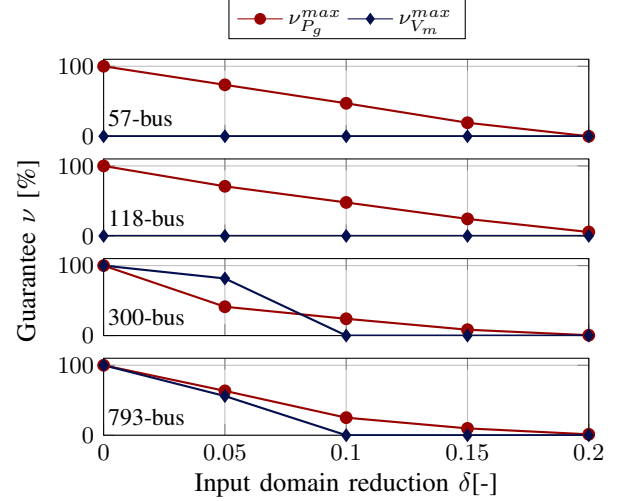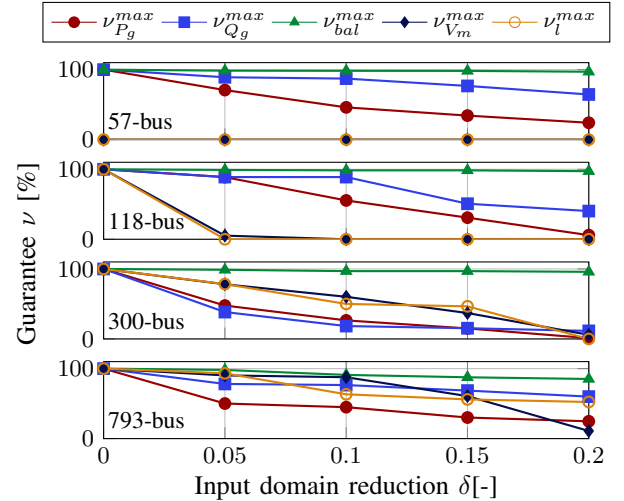
D. Obtain Physically Realizable State

Despite significantly reducing worst-case constraint violations, we were unable to eliminate infeasibilities across the full input domain completely. Hence, direct deployment of the NN is a challenge as feasibility cannot be fully guaranteed.

Table IV summarizes the results obtained by combining the NN outputs with either a feasibility restoration or a warm-start procedure. We compare both computational time and

Fig. 2: Guarantee ν against δ-factor for the Power NN models across the 57-, 118-, 300- and 793-bus systems.

Fig. 3: Guarantee ν against δ-factor for the Voltage NN models across the 57-, 118-, 300- and 793-bus systems.

cost against the AC-OPF solution computed using Power-Models.jl in Julia. For nearly all cases, both the warm-start and feasibility restoration approaches achieved a reduction in computational time relative to the AC-OPF baseline. The warm-started runs generally converged to the same solution as the AC-OPF, with the exception of the 300-bus system, where

TABLE III: Additional Verified Worst-Case Violations Computed for the Voltage NN, Not Possible for the Power NN.

	57-bus		118-bus		300-bus		793-bus	
	Voltage NN		Voltage NN		Voltage NN		Voltage NN	
Violation	base	crown	base	crown	base	crown	base	crown
$\nu_{Q_g}^{max}$ [pu]	10.3	3.52	11.9	1.57	534	26.1	86.0	10.5
ν_l^{max}	1.48	0.00	4.92	0.07	3.19	0.84	311	2.24
$\nu_{V_m}^{max}$ [pu]	0.38	0.00	1.28	0.00	1.17	0.51	1.57	1.38
ν_{bal}^{max} [pu]	0.23	0.23	1.50	1.32	14.1	8.06	12.8	9.17

TABLE IV: Comparison of Warm Start and Feasibility Restoration on 1000 Samples using the Base Neural Networks

| | 57-bus | | | | 118-bus | | | | 300-bus | | | | 793-bus | | | |
| | Warm | | Rest. | | Warm | | Rest. | | Warm | | Rest. | | Warm | | Rest. | |
	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time
PowerModels.jl	$2.7e4	0.18s	$2.7e4	0.18s	$7.0e4	0.32s	$7.0e4	0.32s	$3.8e5	1.25s	$3.8e5	1.25s	$3.4e6	3.34s	$3.4e6	3.34s
Power NN (base)	0.0%	-44.4%	+12.7%	-16.7%	0.0%	+3.0%	+28.4%	-15.6%	-0.2%	-4.8%	+47.2%	-26.4%	0.0%	+3.9%	+185%	-8.7%
Voltage NN (base)	0.0%	-38.9%	+0.6%	-22.2%	0.0%	+3.0%	+4.0%	+9.4%	-0.8%	-6.4%	+9.1%	-21.6%	0.0%	-3.0%	+84%	-7.5%

relaxations were applied to the slack bus constraint.

In terms of cost performance, the suboptimality of the Power NN was consistently higher than that of the Voltage NN. The feasibility restoration for the 793-bus system produced a solution that was notably far from optimal. Interestingly, while the Power NN successfully converged during training for this large system, the Voltage NN had not yet converged after 1 000 epochs. Nevertheless, the Voltage NN trained with worst-case penalties ("crown") achieved a cost suboptimality of 41%, representing a substantial improvement compared to the "base" NN shown in Table IV.

Although the current feasibility restoration procedure requires solving a relatively expensive optimization problem, it could be implemented more efficiently in the future using gradient-based correction steps.

V. CONCLUSION

This work introduced a verification-informed neural network framework for AC optimal power flow (AC-OPF) based on practical linearizations. By incorporating verification feedback into training, the proposed method effectively reduced worst-case constraint violations and improved model robustness. We objectively compared two AC-OPF proxy formulations and evaluated the practicality of implementing worst-case violation penalties during training and their effect on predictive accuracy. The framework was successfully applied to verify large-scale AC-OPF proxies up to 793-buses, demonstrating both scalability and practical feasibility. Future work will focus on fully eliminating worst-case violations and extending verification to include inter-temporal constraints such as generator ramping limits, thereby enabling safe deployment of neural network proxies in real-time optimal control without the need for costly post-processing or feasibility restoration.

REFERENCES

[1] D. K. Molzahn, I. A. Hiskens, *et al.*, "A survey of relaxations and approximations of the power flow equations," *Foundations and Trends® in Electric Energy Systems*, vol. 4, no. 1-2, pp. 1–221, 2019.

[2] R. Nellikkath and S. Chatzivasileiadis, "Physics-informed neural networks for ac optimal power flow," *Electric Power Systems Research*, vol. 212, p. 108412, 2022.

[3] K. Baker, "Learning warm-start points for ac optimal power flow," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, IEEE, 2019.

[4] F. Diehl, "Warm-starting ac optimal power flow with graph neural networks," in *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, pp. 1–6, 2019.

[5] P. L. Donti, D. Rolnick, and J. Z. Kolter, "Dc3: A learning method for optimization with hard constraints," *arXiv preprint arXiv:2104.12225*, 2021.

[6] F. Fioretto, T. W. Mak, and P. Van Hentenryck, "Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 630–637, 2020.

[7] K. Xu, H. Zhang, S. Wang, Y. Wang, S. Jana, X. Lin, and C.-J. Hsieh, "Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers," *arXiv preprint arXiv:2011.13824*, 2020.

[8] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C.-J. Hsieh, and J. Z. Kolter, "Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification," *Advances in neural information processing systems*, vol. 34, pp. 29909–29921, 2021.

[9] Z. Shi, Q. Jin, Z. Kolter, S. Jana, C.-J. Hsieh, and H. Zhang, "Neural network verification with branch-and-bound for general nonlinearities," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 315–335, Springer, 2025.

[10] C. Brix, S. Bak, T. T. Johnson, and H. Wu, "The fifth international verification of neural networks competition (vnn-comp 2024): Summary and results," *arXiv preprint arXiv:2412.19985*, 2024.

[11] A. Venzke, G. Qu, S. Low, and S. Chatzivasileiadis, "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, 2020.

[12] S. Chevalier and S. Chatzivasileiadis, "Global performance guarantees for neural network models of ac power flow," *arXiv preprint arXiv:2211.07125*, 2022.

[13] R. Nellikkath and S. Chatzivasileiadis, "Minimizing worst-case violations of neural networks," *arXiv preprint arXiv:2212.10930*, 2022.

[14] V. Kekatos, G. Wang, H. Zhu, and G. B. Giannakis, "Psse redux: convex relaxation, decentralized, robust, and dynamic solvers," *Advances in Electric Power and Energy: Static State Estimation*, pp. 171–208, 2020.

[15] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C.-J. Hsieh, and J. Z. Kolter, "Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

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

[17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[19] C. Coffrin, R. Bent, K. Sundar, Y. Ng, and M. Lubin, "Powermodels. jl: An open-source framework for exploring power flow formulations," in *2018 Power Systems Computation Conference (PSCC)*, IEEE, 2018.

[20] I. Dunning, J. Huchette, and M. Lubin, "Jump: A modeling language for mathematical optimization," *SIAM review*, vol. 59, no. 2, pp. 295–320, 2017.

[21] L. Thurner, A. Scheidler, F. Schafer, J. H. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun, "pandapower - an open source python tool for convenient modeling, analysis and optimization of electric power systems," *IEEE Transactions on Power Systems*, 2018.

[22] JuliaPy, "PyCall.jl: Calling Python Functions from the Julia Language." https://github.com/JuliaPy/PyCall.jl, 2023. Accessed: 2025-10-D14.

[23] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, 2006.

[24] D. C. Center *et al.*, "DTU Computing Center resources," 2024.

[25] B. Giraud *et al.*, "AC-OPF-Verification." https://github.com/bastiengiraud/AC-OPF-verification, 2025.

[26] Z. Shi, Q. Jin, Z. Kolter, S. Jana, C.-J. Hsieh, and H. Zhang, "Neural network verification with branch-and-bound for general nonlinearities," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2025.