Data-driven Methods for Optimal Power Flow in Smart Grid

Junfei Wang

A DISSERTATION SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN The DEPARTMENT of ELECTRICAL
ENGINEERING & COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

July 2024

Abstract

The modern power grid is undergoing significant changes, driven by increasing complexity, the integration of renewable energy sources, and the urgent need to reduce greenhouse gas emissions. These challenges necessitate more advanced methods for power grid operations. System operators continuously solve the Optimal Power Flow (OPF) problem at regular intervals to determine the most economical dispatch of power while balancing electricity supply and demand. However, traditional OPF and convex relaxation methods often face issues related to feasibility and computational speed. Recently, machine learning methods have gained considerable attention as potential solutions to these challenges. As discussed in the literature, these methods include supervised learning, hybrid approaches that combine physical solvers or equations with machine learning, and unsupervised learning. Despite these advancements, there remain research gaps that need to be addressed. In this thesis, three mechanisms for addressing the OPF problem from different perspectives are proposed. Firstly, a supervised learning algorithm with a subsequent feasibility calibration method is introduced. Secondly, a generative adversarial network (GAN) with a representation learning module is studied and employed as an optimizer for the OPF problem. Lastly, the nearly convex nature of power flow data is investigated, motivating the development of a data-driven convex relaxation approach to solve the OPF problem. This thesis makes significant contributions to the literature by ensuring the feasibility of OPF solvers through post-process algorithms with theoretical support, relaxing the assumption of having optimal solutions for training, and demonstrating high performance of data-driven OPF methods on large systems, such as the PGLIB 2000-bus system.

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Professor Pirathayini Srikantha. My six years in her lab have been immensely enriching, providing me with not only knowledge and experience but also a profound life philosophy. During my journey as a Ph.D. student, her encouragement, guidance, and rigorous training helped shape me into a more professional, skillful, and thoughtful individual.

I am also deeply grateful to Professor Aijun An and Professor Hui Jiang for their help in the realm of machine learning. A special thanks to Professor Priya Donti for inspiring my thoughts on power engineering.

I owe a great deal to my friends Moshi, Parham, and Yifang. Our conversations always sparked novel thoughts and ideas.

Lastly, I would like to express my love and gratitude to my family. To my parents, my wife Mia, and my sons William and Brooklyn, your unwavering support has been my cornerstone. William, who joined us during my Master's studies, and Brooklyn, who arrived during my Ph.D., have both deepened my perspective on life and the world, ultimately benefiting my research profoundly.

Contents

List of Tables

List of Figures

1 Introduction

With the growing concerns of climate change, the increasing penetration of Distributed Energy Resources (DERs), the presence of unpredictable loads, and the proliferation of electric vehicles, the modern electric grid is undergoing a remarkable transition. This energy transition necessitates the adoption of more advanced operation and control methods to ensure grid efficiency and stability.

1.1 Power Grid: Transitions and Challenges

The conventional power grid comprises three core elements: power generation, the transmission network, and the distribution network as shown in Figure 1. Typically, electricity is generated in distant power plants and transmitted at high voltages across the transmission network before reaching end-users via distribution networks. This system has existed as a classic physical infrastructure for a long time, with power flowing uni-directionally from generation to end-users. In operating the power grid, Independent System Operators (ISOs) must continuously balance power demand from users with the supply from generators [2]. Operational failures resulting in imbalances can lead to energy wastage, voltage instability, equipment damage, and even regional blackouts.

However, the impacts of climate change are becoming increasingly evident, with storms, droughts, fires, and flooding growing in strength and frequency [3, 4]. According to the 2018 intergovernmental report on climate change, scientific evidence unequivocally demonstrates that to mitigate the most severe effects of climate change and maintain a habitable planet, we must cap the global temperature increase at 1.5°C above pre-industrial levels. To adhere to the Paris Agreement's call for limiting global warming, emissions must be slashed by 45% by 2030 and achieve net zero by 2050 [5]. Achieving net zero entails reducing carbon

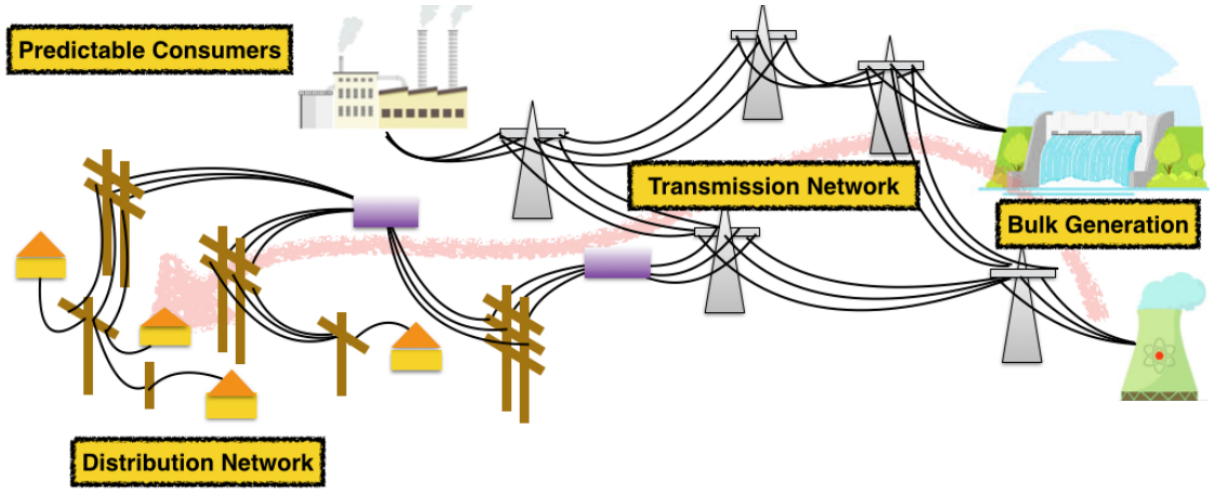

Figure 1: Simplified Structure of the Traditional Grid [1]

emissions to a minimal level of residual emissions, which can be absorbed and securely stored by nature and other carbon dioxide removal measures, thereby leaving zero emissions in the atmosphere [6].

Against this backdrop, the emergence of the 'smart grid', initially proposed in 2007 [7, 8], represents a paradigm shift. This system, as shown in Figure 2, integrates various elements such as DERs like solar and wind power, energy storage solutions, advanced sensing and measurement technologies, information and communication technologies, and analytical and decision-making tools. Notably, smart grid consumers differ from traditional ones, with residential properties equipped with solar photovoltaic generators and electric vehicles capable of injecting power into the grid. Despite leveraging existing infrastructure, smart grids boast distinct power resources, consumer profiles, and communication pathways, transforming into bi-directional cyber-physical systems.

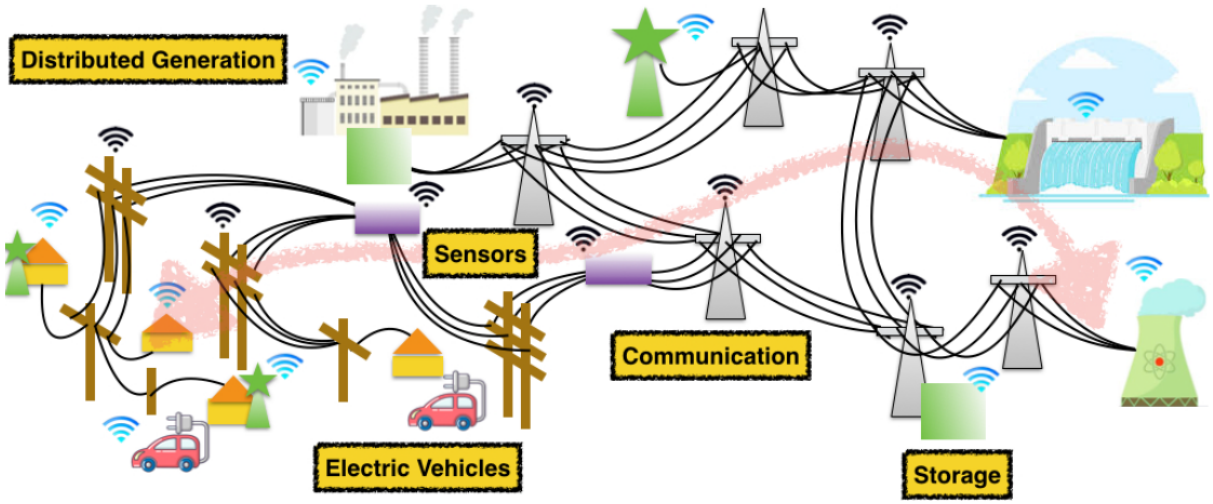

Figure 2: Simplified Structure of the New Grid [1]

1.2 Power Grid Operations in North America

The North American electric power grid, one of the world's most complex and extensive systems, comprises over 30,000 generators (including those powered by natural gas, coal, nuclear, hydroelectric, wind, and solar), spans more than 400,000 kilometers of high-voltage (greater than 230 kV) transmission lines, and serves over 165 million customers [9, 10]. Grid operations are managed by a combination of Independent System Operators (ISOs) and Regional Transmission Organizations (RTOs) in North America.

These entities are responsible for coordinating, controlling, and monitoring the operation of the electrical power system to ensure reliability and efficiency. The nine key ISOs and RTOs include: ISO New England (ISO-NE), Midcontinent Independent System Operator (MISO), New York Independent System Operator (NYISO), PJM Interconnection, Electric Reliability Council of Texas (ERCOT), California Independent System Operator (CAISO), Southwest Power Pool (SPP), Alberta Electric System Operator (AESO) and Ontario Independent Electricity System Operator (IESO) [11]. Their geographic distribution is shown in the map in Fig.3.

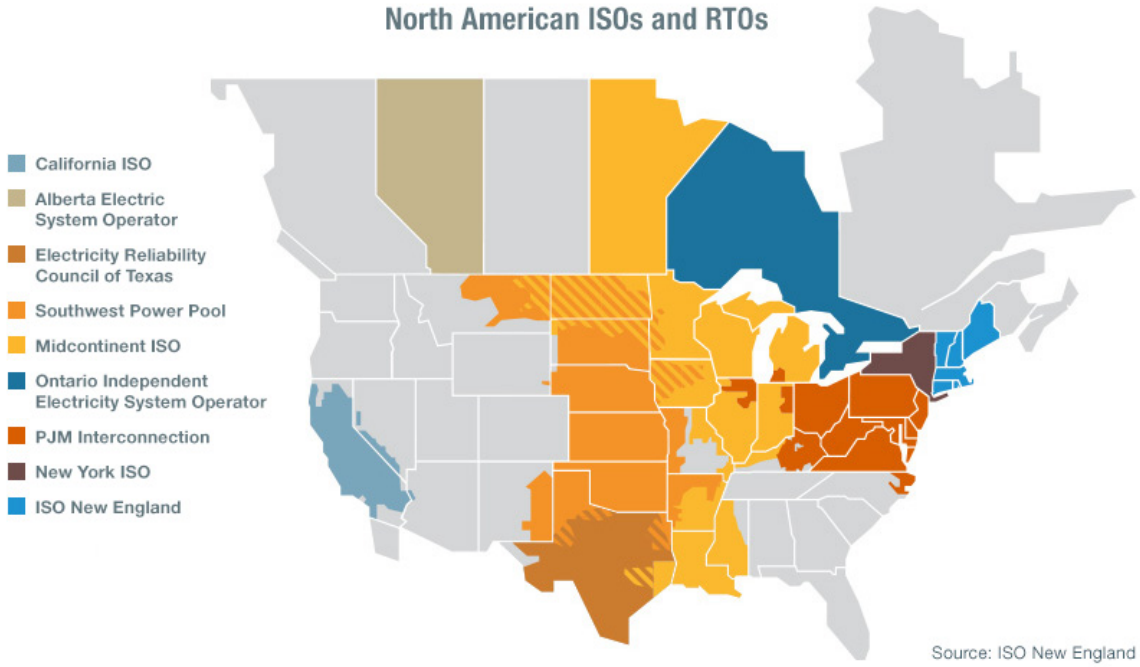

Figure 3: Geographic Distribution of ISOs and RTOs in Northe America [12]

In real-time market operations, ISOs continuously solve the Optimal Power Flow (OPF) problem at regular intervals to determine the most economical dispatch of power while balancing electricity supply and demand, such as the California ISO, which runs OPF every five minutes [13]. We use the two terms OPF and ACOPF (Alternating Current Optimal Power Flow) interchangeably in this thesis. OPF algorithms are used to determine the most efficient dispatch of generating units while considering various constraints such as transmission limits, generator capacities, and voltage stability. However, efficiently solving OPF problem has remained a longstanding challenge in power engineering due to its nonconvex nature, which has been proven to be NP-hard [14] and cannot be solved within an acceptable computational time. It has been reported that in the U.S. alone, a 5% increase in market efficiency with better OPF solvers can save approximately 6 billion dollars per year [2]. Moreover, the recent endeavors of grid modernization and accommodating high penetration of DERs pose a computational challenge of solving OPF problems. Specifically, increasing DERs (e.g., wind

farm and solar photovoltaic) introduces significant uncertainty into the load/generation, requiring operators to solve OPF problems for numerous scenarios repeatedly and promptly. Therefore, the complexity of the OPF problem and the emergence of new challenges in the modern grid underscore the need for OPF solvers at finer time scales.

1.3 Problem Statements and Research Objectives

The challenges stemming from the paradigm shift in the modern power grid and the need for more advanced operational tools present rich opportunities for research into reliable optimization algorithms in power systems. As renewable energy sources, distributed generation, and smart grid technologies become more prevalent, the complexity of the power grid increases. This complexity introduces significant challenges in maintaining grid stability, efficiency, and reliability. The ACOPF problem, which is essential for determining the most efficient operating conditions for the power grid, is particularly challenging due to its nonlinear and nonconvex nature, as well as the numerous hard constraints that must be satisfied.

Traditional solvers, such as the Interior Point Method (IPM) [15] and Semidefinite Programming (SDP) [16], struggle to solve the problem within acceptable computational times (e.g., 5 or 15 minutes). These methods often fail to provide timely solutions for real-time applications, which are crucial for the dynamic and fast-changing conditions of modern power systems. As a result, there is a pressing need for innovative approaches that can deliver fast and reliable solutions.

In this thesis, the potential of leveraging data-driven techniques is explored to overcome the limitations of traditional solvers. By utilizing the power of data under different assumptions in the ACOPF problem, such as using a traditional solver to generate labelled data or only accessing historical operational data, three fast ACOPF solvers are proposed form

different perspectives. These solvers incorporate advanced machine learning algorithms and optimization techniques to enhance computational efficiency and accuracy.

Chapter 2 provides a comprehensive background on power systems, power flow studies, and the formulation of the ACOPF problem. This section lays the foundation for understanding the complexities and requirements of the ACOPF problem in modern power grids. Chapter 3 reviews the existing literature on solving the ACOPF problem through various approaches, including relaxation methods, approximation techniques and data-driven approaches. This review highlights the strengths and limitations of current methodologies and identifies gaps that our research aims to address.

Chapters 4, 5, and 6 introduce three novel optimization algorithms. Chapter 4 presents a physics-informed machine learning approach that integrates domain knowledge with data-driven techniques to improve the output solutions. Chapter 5 discusses an unsupervised generative learning method that leverages historical data to model the true data distribution and solve the ACOPF problem in this distribution. Chapter 6 introduces a convex neural network framework that provides a fast convex optimization solver. Finally, the thesis is concluded and the potential future research directions are discussed in Chapter 7.

2 Background

2.1 Power Grid

In power engineering, key components such as generation sectors, load points, and substations are referred to as buses. These buses are interconnected by transmission lines, including both high-voltage and distribution lines. As a result, the power grid can be effectively represented as a graph, with nodes corresponding to buses and edges corresponding to transmission lines.

Consider the PJM 5-bus system [17] as an example in Figure 4, the system consists of 5 buses(A-E), and 6 lines. Each node connects to either generators, or loads, or both. Each

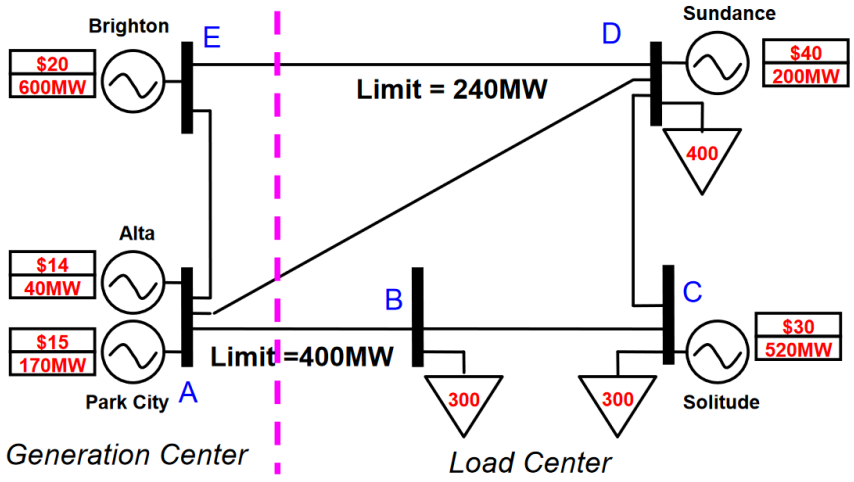

Figure 4: PJM 5-Bus System [17]

transmission line has admittance denoted as y_{ik} in which i and k are two buses the line connected. The admittance represents how easy the current flows through a transmission line between bus i and bus k. It is the reciprocal of the line's impedance Z_{ik}, which can be calculated as follows in Equation.1. In the equation, R_{ik}, X_{ik}, G_{ik} and B_{ik} are resistance,

reactance, conductance and susceptance, respectively.

$$\begin{aligned} y_{ik} &= \frac{1}{Z_{ik}} = \frac{1}{R_{ik} + jX_{ik}} \\ &= G_{ik} + jB_{ik} \end{aligned} \tag{1}$$

For simplicity in power flow studies, the nodal admittance matrix, also known as the Y-bus matrix, is often used instead of individual impedance or admittance values for each transmission line. The Y-bus matrix not only represents the network's topology but also includes the line admittance and the shunt admittance at each bus. This makes it a powerful tool for analyzing the steady-state operation of the power system. By assembling the Y-bus matrix, we can simplify the power flow analysis and other related calculations, making it easier to study and optimize the operation of the power grid.

Definition 1. *The nodal admittance matrix (Y-bus matrix) for a grid with N buses is a N-by-N matrix [18]*

$$Y_{bus} = \begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1N} \\ Y_{21} & Y_{22} & \cdots & Y_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{N1} & Y_{N2} & \cdots & Y_{NN} \end{bmatrix}$$

in which $Y_{ik} = \begin{cases} y_i + \sum_{j=1, j\neq i}^{N} y_{ij} & \text{if } i = k \\ -y_{ik} & \text{if } i \neq k \end{cases}$, *$y_{ij}$ and y_{ik} are line admittance, and y_i is shunt admittance.*

2.2 Power Flow Studies

The power flow problem is a fundamental analysis task in power engineering. It involves determining the voltage, current, and power flows within an electrical network under a given load and generation scenario.

2.2.1 Buses and Variables

In power flow studies, there are three sets of buses: generator buses (PV buses, denoted as set $\mathcal{G}$), load buses (PQ buses, denoted as $\mathcal{L}$), and slack buses (reference buses, typically indexed as 0). Each bus has four quantities including active power, reactive power, voltage magnitude, and voltage angle. Generator buses $g \in \mathcal{G}$ control the active power generation P_{G_g} and voltage magnitude $|V_{G_g}|$, load buses $i \in \mathcal{L}$ consume both active power P_{D_i} and reactive power Q_{D_i}, and the slack bus with fixed voltage magnitude $|V_0|$ and phase angle ϕ_0 balances the system by providing or absorbing the difference between generated and consumed power. These are tabulated in Table 1. The goal of the load flow study is to find all of them at each bus in a given grid based on power flow equations. These are the basis for calculating optimal power flow because all power flow solutions, whether optimal or not, need to be feasible concerning power flow equations at all buses.

Table 1: Variables in Power Flow Studies

Type of Bus\Variables	**Known Variables**	**Unknown Variables**	**Scope**
Slack	$\|V_0\|$,ϕ_0	P_0,Q_0	Reference Bus
Load	P_{D_i},Q_{D_i}	$\|V_{D_i}\|$,ϕ_{D_i}	$i \in \mathcal{L}$
Generator	P_{G_g},$\|V_{G_g}\|$	Q_{G_g}, ϕ_{G_g}	$g \in \mathcal{G}$

2.2.2 Bus Injection Model

The complex power flowing from bus i to bus k is denoted by S_{ik} and it can be decomposed into its active power P_{ik} and reactive power Q_{ik} components:

$$S_{ik} = P_{ik} + jQ_{ik} \tag{2}$$

It can also be described as the product of voltage V_i and the conjugate of current I_{ik}^*:

$$S_{ik} = V_i I_{ik}^* \tag{3}$$

According to Ohm's Law, the current is directly proportional to the potential difference and inversely proportional to the impedance of the circuit. Because admittance is the reciprocal of impedance, we have:

$$I_{ik} = y_{ik}(V_i - V_k) \tag{4}$$

Using the Y_{bus} matrix in Definition.1, we have:

$$I = \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_N \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1N} \\ Y_{21} & Y_{22} & \cdots & Y_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{N1} & Y_{N2} & \cdots & Y_{NN} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_N \end{bmatrix} = Y_{bus} V \tag{5}$$

where I_i is the current flowing into the node i, and V_i is the voltage at the node i. For a

specific node i, the injected current can be further rewrite into the following form:

$$\begin{aligned} I_i &= Y_{i1}V_1 + Y_{i2}V_2 + \cdots + Y_{N1}V_N \\ &= V_i(y_i + \sum_{k=1,k\neq i}^{N} y_{ik}) - \sum_{k=1,k\neq i}^{N} V_k y_{ik} \\ &= V_i y_i + \sum_{k=1,\ldots,N;k\neq i} y_{ik}(V_i - V_k) \end{aligned} \tag{6}$$

Recalling Kirchhoff's Law, stating that for any node (junction) in an electrical circuit, the sum of currents flowing into that node is equal to the sum of currents flowing out of that node. This motivate the bus injection model in power flow study as follows:

Definition 2. *The bus injection model states that the power flowing into any bus i must be equal to the power going out from that bus. Specifically, this means that the sum of power generation S_{G_i} and the power flowing into bus i equals the sum of the power withdrawn S_{D_i} and all power flowing out from bus i [16].*

This model can be mathematically defined in Equation 7, where θ_{ik} is the angle of Y_{ik}.

$$S_{G_i} - S_{D_i} = V_i \sum_{k=1}^{N} Y_{ik}^* V_k^* \tag{7}$$

Because complex power is the combination of active and reactive power as denoted in Equation.2, so Equation.7 can be decomposed into active power flow equation.8 and reactive power flow equation.9.

$$\begin{aligned} P_{G_i} - P_{D_i} &= Re(V_i \sum_{k=1}^{N} Y_{ik}^* V_k{}^*) \\ &= |V_i| \sum_{k=1}^{N} |V_k| \left(G_{ik}\cos(\phi_i - \phi_k) + B_{ik}\sin(\phi_i - \phi_k)\right) \end{aligned} \tag{8}$$

$$
\begin{aligned}
Q_{G_i} - Q_{D_i} &= Im(V_i \sum_{k=1}^{N} Y_{ik}^* V_k^*) \\
&= |V_i| \sum_{k=1}^{N} |V_k| \left(G_{ik} \sin(\phi_i - \phi_k) - B_{ik} \cos(\phi_i - \phi_k)\right)
\end{aligned}
\tag{9}
$$

2.2.3 Traditional Solvers for Power Flow Problem

Algorithms for solving the power flow problem have been extensively studied and developed over the years, with several well-established methods in use, such as Gauss-Seidel (GS) algorithm [18] and Newton-Raphson algorithm [19]. GS algorithm uses different ways to update variables for each type of bus and runs these updates in epochs until convergence is achieved. Here are the update rules for the different buses in each epoch:

- **Load Buses:** As shown in Table 1, power demand P_{D_i} and Q_{D_i} are given on load buses, so voltages are updated by the following rule:

$$
V_i^{e+1} = \frac{1}{Y_{ii}} \left[\frac{-P_{D_i} + jQ_{D_i}}{V_i^{e*}} - \sum_{k=1, k \neq i}^{N} V_k^e Y_{ik} \right] \tag{10}
$$

  where i is the index of the target bus, e is the epoch of updates across the system, and Y is the admittance matrix.

- **Generator Buses:** The active power generation P_{G_i} and voltage magnitude $|V_i|$ are fixed on each generator bus $i \in \mathcal{G}$. As such, the algorithm computes the reactive power Q_{G_i} in the following way:

$$
Q_{G_i}^{e+1} = Im\left(V_i^e \sum_{k=1}^{N} Y_{ik}^* V_k^{e*} \right) + Q_{D_i} \tag{11}
$$

Then, the voltages can be calculated via:

$$V_i^{e+1} = \frac{1}{Y_{ii}} \left[\frac{P_{G_i} - P_{D_i} - j(Q_{G_i}^{e+1} - Q_{D_i})}{V_i^{e*}} - \sum_{k=1, k \neq i}^{N} V_k^{e+1} Y_{ik} \right] \tag{12}$$

Due to the voltage magnitude being fixed, only phase angles in V_i^{e+1} are used.

- **Slack Bus:** This bus is used as a reference bus in the system and absorbs any imbalance of active and reactive power. These can be updated via Equations 8 and 9.

The GS algorithm proceeds by sequentially updating each bus voltage using the most recent estimates, progressively reducing the discrepancy between the calculated and specified power injections. Convergence is achieved when the updates to the bus voltages fall below a predetermined tolerance level, indicating that the power flow solution is sufficiently accurate. The primary advantage of the GS algorithm lies in its simplicity and relatively low computational requirements for small to moderately sized systems. While the GS algorithm is relatively easy to implement and understand, it can be slow to converge for large systems. Nevertheless, it serves as a valuable foundational method in power system analysis.

2.3 Optimal Power Flow Problem

The concept of Optimal Power Flow (OPF) was first proposed in the 1960s [20], and it has since become one of the most essential and widely studied topics in power system operations. The primary motivation for OPF arose from the need for system operators to determine the most efficient way to dispatch available generators to meet the grid's demand at minimal cost. Also, many optimization problems in power system can be transformed into OPF problem, such as Economic Dispatch, Voltage Control and Congestion Management. Additionally, OPF can connect to power market and decide electricity prices [21].

OPF is a constrained optimization problem to minimize a specific goal, e.g., the financial cost of the power generation, power loss in the system and/or green house emissions, subject to physical constraints, such as the power flow balance constraints in Equations 8 and 9, and operational constraints regarding the capacities of generators, voltage, and branch flow constraints, etc. Thus, output solutions should have two properties: the first one is feasibility which means they satisfy all equality and inequality constraints; the second property is the optimality, which guarantees the omitted solutions with the lowest value of the cost function.

Definition 3. *The standard OPF problem with bus injection model in Definition.2 is in the following form:*

$$\mathcal{P}_{ACOPF} : \min_{P_{G_g}} \sum_{g \in \mathcal{G}} C_g(P_{G_g})$$

$$s.t.\ \forall\ g \in \mathcal{G} : P_{G_g}^{min} \leq P_{G_g} \leq P_{G_g}^{max} \qquad \textbf{[C1]}$$

$$Q_{G_g}^{min} \leq Q_{G_g} \leq Q_{G_g}^{max} \qquad \textbf{[C2]}$$

$$s.t.\ \forall\ i \in \mathcal{N} : P_{G_i} - P_{D_i} + j(Q_{G_i} - Q_{D_i}) = V_i \sum_{j \in \mathcal{N}} V_j^* Y_{ij}^* \qquad \textbf{[C3]}$$

$$V_i^{min} \leq |V_i| \leq V_i^{max} \qquad \textbf{[C4]}$$

$$\phi_i^{min} \leq \phi_i \leq \phi_i^{max} \qquad \textbf{[C5]}$$

$$|V_i(V_i^* - V_j^*)Y_{ij}^*| \leq S_{i,j}^{max}\ \forall\ j \in \mathcal{N}, j \neq i \qquad \textbf{[C6]}$$

where $\mathcal{N}$ is the set of all buses, and $\mathcal{G}$ is the set of power generators. The objective of this formulation is to compute the optimal setpoints for the generation systems in the network while ensuring that all system constraints are met. In $\mathcal{P}_{ACOPF}$, P_{G_g} and Q_{G_g} are the real and reactive power generated by generator $g \in \mathcal{G}$ that can be dispatched, while P_{D_i} and Q_{D_i} are the real and reactive power demands of bus $i \in \mathcal{N}$. The cost of real power injected

by generator g, denoted as C_g, is typically a quadratic function [22]. Constraints **[C1]** and **[C2]** specify the upper and lower limits of real and reactive power generation by each generator. Constraint **[C3]** enforces power balance at each bus. Constraints **[C4]** and **[C5]** impose limits on the voltage magnitude and phase angles of buses, respectively. Constraint **[C6]** reflects the branch flow limits. The inputs to $\mathcal{P}_{ACOPF}$ are the real and reactive power demands of each bus (P_{D_i} and Q_{D_i}), which vary over short time intervals (e.g., seconds) to reflect changes in demand and renewable generation. The outputs include the real and reactive power generated by each generator, as well as the bus voltage magnitudes and phase angles for all buses.

This OPF problem has been proven to be NP-Hard in the worst case [14], due to the non-convex and non-linear nature of constraints **[C3]**, **[C4]** and **[C6]**. Furthermore, with the recent modernization of the grid, the network may have a high penetration of DERs such as wind farms and solar photovoltaic. The uncertainties and complexities of the OPF problem are thus escalated, posing a significant computational challenge for real-time solutions. Traditionally, OPF is solved at fixed intervals, such as hourly 15 minutes, or 5 minutes [2], assuming that power demand within these intervals change only slightly. However, this assumption is increasingly impractical in the modern power grid with its growing population of DERs and unpredictable power usage. Therefore, effectively and efficiently solving OPF in this more complex and dynamic environment in real-time is a crucial research topic in power engineering.

3 Literature Review

There is a large body of research on ACOPF problems. This section reviews some important works in the literature and discusses their advantages and drawbacks. In Section 3.1, convex relaxation of the original problem is illustrated. Subsequently, the interior point methods are discussed in Section 3.2. Then, three types of machine learning based optimization techniques are reviewed including supervised learning for ACOPF in Section 3.3, hybrid learning in Section 3.4, and unsupervised learning in Section 3.5. Finally, a summary and research gap are proposed in Section 3.6.

3.1 Convex Relaxations

In practice, we often encounter highly complex constrained optimization problems that may take an unacceptably long time to find global optimal solutions. Convex relaxation is a technique used to transform the original non-convex problem into a convex one by relaxing some of the constraints. This transformation makes the relaxed problem much easier to tackle. This section introduces three re

3.1.1 Direct Current Optimal Power Flow

One of the most successful convex relaxation techniques for ACOPF is Direct Current Optimal Power Flow (DCOPF) [23–28]. It is widely used in current power system operations due to its convexity and computational benefits, which allow it to be solved on fast timescales. DCOPF simplifies the original problem by making the following three assumptions:

- Line resistances are negligible compared to line reactances ($R_L << X_L$, for all lines). This assumption implies that grid losses are neglected and line parameters are simpli-

fied.

- The difference between adjacent bus voltage angles is small.
- Magnitudes of bus voltages are set to 1.0 per unit (flat voltage profile).

The ACOPF problem in Definition.3 can be simplified as the DCOPF form:

$$\begin{aligned}
&\min_{P_g^{gen}} \sum_{g \in \mathbb{G}} C_g(P_g^{gen}) \\
&s.t.\ P_{G_i} - P_{D_i} = \sum_{k=1}^{N} B_{ij}(\phi_i - \phi_k)\, \forall i \in \mathcal{N} \\
&\quad P_{G_g}^{min} \leq P_{G_g} \leq P_{G_g}^{max},\ \forall g \in \mathcal{G} \\
&\quad \phi_i^{min} \leq \phi_i \leq \phi_i^{max},\ \forall i \in \mathcal{N}
\end{aligned}$$

DCOPF is a quadratic programming approximation of the original problem with linear constraints and a quadratic cost function. It is convex due to the objective function's positive second-order derivatives and linear constraints. However, assuming fixed voltage magnitudes and ignoring reactive powers are not applicable for real practice. It is shown that the set of points within the feasible region of the DCOPF problem and the set of points within the feasible region of the ACOPF problem have an empty intersection. It means solutions of the DCOPF is never AC feasible [29]. In practice, many grid operators calculate DCOPF setpoints and calibrate these until they are AC feasible in a "DCOPF with AC Feasibility" iterative procedure [30], but research study on carbon emissions [31] shows the DCOPF with AC feasibility may have as much as 28% emissions than ACOPF solutions.

3.1.2 Semi-definite Programming

ACOPF can be converted into a quadratically constrained quadratic programming (QCQP) problem in which both the objective function and the constraints are quadratic functions [32]. Then, it can be relaxed to a semidefinite programming (SDP) problem, and becomes an easier problem solved by convex optimization tools [16, 33, 34]. The OPF in Definition.3 can be expressed in the form of a nonconvex QCQP problem. Thereafter, a standard semidefinite relaxation on the rank transform problem to be a convex relaxation problem [14]. SDP relaxation is designed to obtain a convex supersets of the original problem and minimize the same cost function over these supersets. Due to the tightness of this relaxation, it provides a lower bound to OPF. As shown in the comparative study in Section 6, the output of SDP relaxation commonly violates inequality constraints on voltage magnitude and reactive power generation, as well as the equality constraints on power flow equations. Furthermore, instead of solving for N voltages as defined in Definition 3, SDP solves for N^2 variables. Consequently, the computational complexity increases exponentially with the size of the grid, and SDP may not converge in an acceptable time for large grids.

3.1.3 Second Order Cone Programming

Compare to SDP relaxation, Second Order Cone Programming(SOCP) relaxation of ACOPF have less computational complexity, and the convergence performance is more stable [35–38]. Based on the Branch Flow Model [16], square of voltages and current are defined as variables in the new problem, so phase angles are not in the formulations. Furthermore, SOCP relaxes the one set of equality constraints as inequality constraints to transform the problem as convex optimization problem. However, SOCP relaxation modifies equality constraints to inequality constraints, and drop phase angles in the formulation, which makes the relaxation inexact in transmission networks and some cases in distribution networks [39].

3.1.4 Comparison of Convex Relaxation Techniques

The complexity of the three algorithms discussed in this section can be ranked in descending order as: SDP relaxation, SOCP relaxation, and DCOPF. Additionally, SDP is the tightest relaxation, SOCP is second, and DCOPF is often categorized as an approximation algorithm rather than relaxation.

3.2 Interior Point Methods

Interior Point Methods (IPM) [40–43] solve the OPF problem by traversing the interior of the feasible region, avoiding the boundaries defined by the constraints until convergence is achieved. This approach iteratively solves a sequence of approximations to the original nonlinear problem. At each iteration, the algorithm solves a system of linear equations derived from the Karush-Kuhn-Tucker (KKT) conditions of the problem, and refine the current solution until it converge to an optimum. IPM serves as the algorithm to generate labelled data for supervised learning-based OPF methods such as [45, 52] and acts as the benchamark result regarding the optimality [67, 68]. However, IPM can be computationally intensive, especially for very large-scale power systems [44]. Each iteration involves solving large systems of linear equations, which can be time-consuming and require significant computational resources. Additionally, IPM may have difficulties in scenarios with poorly conditioned problems or when the initial feasible point is far from the optimal solution.

3.3 Supervised Learning for OPF

Supervised learning methods [45–49] treat ACOPF solver as a black-box, which can be learned from training dataset with input and optimal solution pairs.

Reference [45] proposes a supervised learning model which takes the power demand at each bus as input and uses four neural networks to predict voltage magnitude, phase angle, active power generations, and reactive power generations, respectively. It states that the results of purely supervised learning models violate both equality and inequality engineering constraints without any additional regularization. Therefore, their work combines deep learning and Lagrangian optimization to not only optimize the regular machine learning cost but also minimize the penalty of physical constraints. This approach can be seen as a minimax game between the neural network and the dual variables of the Lagrangian. The equilibrium point is approximated by alternatively doing stochastic gradient descent and ascent on the network's parameters and dual variables.

In reference [46], classification models including Naive Bayes, Logistic Regression, and Decision Trees, as well as regression models including Linear Regression and Gaussian Process Regression, are trained to predict the optimal cost and solvability of ACOPF respectively. Nevertheless, ISOs desire the complete solutions of the best dispatch, rather than just the best cost and its existence. This can still be used as a solvability check and benchmark cost for other ACOPF solvers.

Considering the small errors in the input of the ACOPF problem, a robust ACOPF solver is proposed in [47]. This method is a multi-layer perceptron variant, which consists of a Denoising Autoencoder (DAE) in each layer. In this setting, the input of the DAE will first be perturbed as $\tilde{X}$ and then encoded to a representation $a = e_\theta(\tilde{X})$ in the bottleneck layer, and subsequently reconstructed back to the input space as $Z = d_\theta(a)$. The goal of the DAE is to use Z to recover X, i.e., the cost function $L_H(X, Z)$ is the mean square error function. By doing this, the representation a will be robust to small errors or changes of the input. SDAE discards the decoder part d_θ, and only use the representations in each layer of the proposed neural network. The output of the model is made up of operation cost, node

voltage of all nodes, generator outputs, flow in all branches. However, there is no guarantee in this method regarding the adherence to neither the equality constraints nor inequality constraints defined in Definition.3.

Reference [48] divides the learning task of ACOPF into three stages to reduce the learning difficulty of each subproblem. In the first stage, the model takes the active and reactive power demand, P_D and Q_D, as input and predicts the optimal branch flow. Subsequently, the second stage calculates the voltage magnitudes $|V|$ and phase angles ϕ based on the output from stage one. The final stage produces the optimal power generations P_G and Q_G. In this work, clustering algorithms are used to categorize training examples into multiple different regions, and then multiple sets of these models with three stages are built to solve the ACOPF problem in each sub-region.

By leveraging the power of both supervised learning and Graph Theory, a Graph Neural Network(GNN) based ACOPF solver is proposed in [49]. It defines a graph signal at each node as:

$$\mathbf{X} = \begin{bmatrix} |V| & \phi & P & Q \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_N^\top \end{bmatrix} \in \mathbb{R}^{N \times 4}.$$

in which $|V|$ are voltage magnitudes, ϕ are phase angles, P and Q active and reactive net power injections respectively. A GNN is a nonlinear map $\Phi(\mathbf{X}; \mathcal{H}, \mathbf{W})$ that is applied to the input $\mathbf{X}$ and takes into account the underlying graph $\mathbf{W}$ and its coefficients $\mathcal{H}$. It consists of a cascade of L layers, each of them applying a graph convolution followed by a nonlinear activation function. Research [21] shows that when the nodal features exhibit a graph-based locality property or topology dependence, the GNN architecture is known to efficiently incorporate the underlying graph embedding. Nonetheless, this assumption commonly does

not hold in power systems due to differences in load, generation, and network topology.

In contrast to the end-to-end learning approaches discussed above, a machine learning model is proposed in [50] to emulate iterative optimization solvers, such as the Matpower Interior Point Solver (MIPS). Instead of using Lagrangian functions or forming Hessian matrices, this work employs a deep learning model $F_R(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ that takes x^k as input and provides x^{k+1} as output, i.e., $x^{k+1} = F_R(x^k)$, until the iterative process converges.

3.4 Hybrid Learning

3.4.1 Hybrid Model of Supervised Learning and Traditional Solvers

To guarantee the feasibility of the output solutions from proposed solvers, recent research works [51–54] separate the variables in ACOPF solutions into independent and dependent variables. Independent variables are predicted by trained machine learning models, while dependent variables are calculated by traditional power flow solvers such as NR or GS [55] solvers. The typical way of this separation [51,52,54] is based on the different types of buses, as shown in Table 1. ConvOPF [53] defines independent variables are the active power generation P_G and reactive power generation Q_G, and then treat generator buses as P/Q buses to solve via traditional power flow solvers. The system model of these algorithms can be shown in Figure 5, where the input of the neural network is active and reactive power load on each load bus, and its outputs are all independent variables.

These algorithms only need to learn a partial mapping instead of the entire mapping from the inputs to complete OPF solutions. Furthermore, integrating traditional power flow solvers guarantees the feasibility of the solutions when they exist. Because the solution of the ACOPF is often on the feasible boundary of the optimization problem, reference [54] adds a tuning parameter to this framework to ensure the output of machine learning models

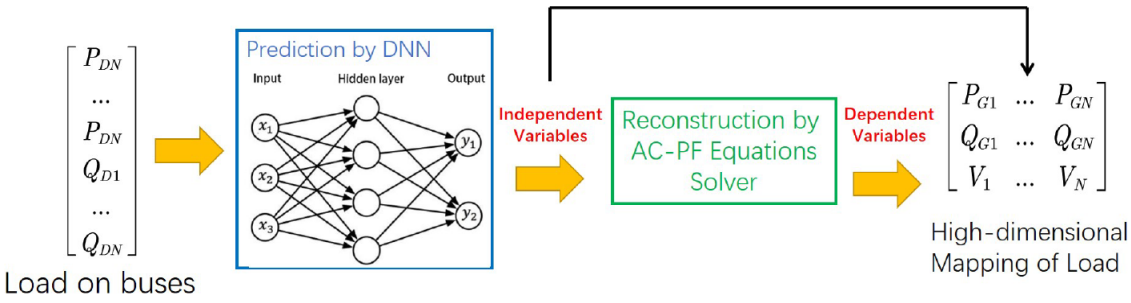

Figure 5: The System Diagram of DeepOPF [52]

is strictly within the interior of the voltage magnitude feasibility set.

In contrast to the hybrid methods in [51–54], the machine learning models in [56, 57] outputs complete solutions with all variables in the ACOPF problem. Subsequently, it uses these solutions as warm-start points for traditional ACOPF solvers. Compared to the default heuristics used in conventional optimization methods, an accurate initial point could theoretically reduce the number of iterations required to reach the optimal point [58].

3.4.2 Hybrid Model of Supervised Learning and Power Flow Equations

The second type of hybrid model [59–62] for ACOPF defines voltage magnitudes $|V|$ and phase angles ϕ as independent variables learned by machine learning models, and then runs power flow equations defined in Equations 8 and 9 for the dependent variables, including active power generation P_G and reactive power generation Q_G. Unlike the first type of hybrid models, the RHS of the power flow equations are all known, making the calculation of the LHS straightforward.

A learning framework is proposed in [59] based on multi-input and multi-output random forest. A random forest model consists of many decision trees that make predictions separately, which are then averaged to minimize error. The trees are base learners and near-independent, reducing the risk of biased decisions or overfitting.

ConvOPF-DOP [60] analyzes the training dataset with clustering algorithms to find operational patterns; for example, the IEEE 30-bus system has 4 patterns. The input layer of the proposed Convolutional Neural Network comprises active power demand P_D, reactive power Q_D, and the label of the pattern in which the power demand lies. The model outputs the optimal voltage magnitude $|V|$ and phase angle ϕ. Subsequently, the algorithm runs power flow equations to obtain optimal active power dispatch P_G and reactive power dispatch Q_G.

DeepOPF-V [61] uses two separate neural networks to predict optimal voltage magnitude $|V|$ and phase angle ϕ. It also suggests that for large power systems, all buses can be split into groups, each predicted by separate Machine Learning models. This reduces the size of the models and significantly limits the increase in training time.

A Sensitivity-informed Deep Neural Networks (SI-DNNs) method is proposed in [62], where the machine learning model is trained to emulate both optimal voltages, including magnitude and phase angle, and the Jacobian matrix of the solution. SI-DNNs achieve the same prediction performance with fewer training data, roughly only 1/10 to 1/4 of the training data. This improvement in sample efficiency reduces the time needed for generating training datasets, which is beneficial for delay-critical power systems applications.

3.5 Unsupervised Learning for OPF

Unsupervised learning commonly means using machine learning algorithms to discover hidden patterns in data without the cumbersome effort of labelling them. In ACOPF problem, collecting a huge optimal solution dataset is computationally expensive and not always realistic. In contrast, historical operational data are easily accessed in the utilities' dataset. Unsupervised learning can be a promising direction to explore historical data and use machine learning for optimization tasks.

DeepOPF-NGT(No Ground Truth) [63] aims to learn the mapping between load configurations (active power demand P_D and reactive power demand Q_D) and the optimal bus voltages (magnitude $|V|$ and ϕ). Then the rest of the solutions are calculated based on power flow Equations 8 and 9. The training data are not needed in this method. Instead, P_D and Q_D are randomly sampled within a 10% variation range of the default load of the bus grid, and the optimization is conducted by directly adding the penalization term for the fuel cost onto the loss function. The Kron reduction method is used in this work to overcome the unobservable nodes in the grid. The diagram of this algorithm is depicted in Figure 6.

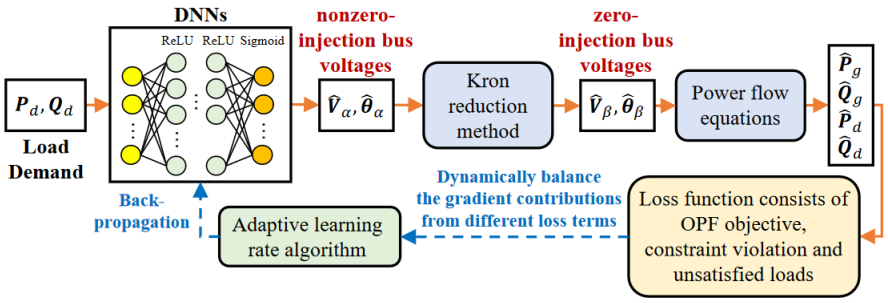

Figure 6: The System Diagram of DeepOPF-NGT [63]

This method is easy to implement and understand. Nevertheless, the objective curve is probably very complex with many local optima, so optimizing the OPF problem by stochastic gradient descent is very likely to be stuck in local optima. In fact, the optimization of the machine learning process accepts local optimal solutions, but OPF does not.

Similar to this idea, [64] adopts GNN as the machine learning model. It defines the input signal to GNN as:

$$\mathbf{Z} := \begin{bmatrix} \text{Rect}(s^d) & \text{Rect}(s^g_{\min}) & \text{Rect}(s^g_{\max}) & v_{\min} & v_{\max} \end{bmatrix} \tag{13}$$

where $\text{Rect}(z) = \begin{bmatrix} \text{Re}(z) & \text{Im}(z) \end{bmatrix}$ represents complex vectors from $\mathbb{C}^N$ as a real matrix in

$\mathbb{R}^{N\times 2}$ containing the real and imaginary components. Similarly, let $\mathbf{X} \in \mathbb{R}^{N\times 4}$ be a graph signal describing the outputs of the OPF problem:

$$\mathbf{X} := [\text{Rect}(s^g) \quad \text{Rect}(v)] \tag{14}$$

Then, a penalty composed of the cost of ACOPF and weighted violation of equality and inequality constraints is applied as the training loss of the GNN. However, experiments show that at least one constraint is violated in 75% of the testing cases for small grids, e.g., the IEEE bus 30 grid.

DC3 [65, 66] uses a trick called implicit layers to transform power flow solvers to be differentiable. Assuming the equality constraint of the original problem can be written in the form of $h_x(y) = 0$, where x is the input of the neural network and y is the solution. The algorithm produces a part of the solution z, and then uses Newton's method or the Gauss-Seidel method to derive the rest part $\phi(z)$ that satisfies $h_x([z^T, \phi(z)^T]^T) = 0$. Taking the derivative of this function with respect to z, we get:

$$\begin{aligned} 0 = \frac{d}{dz} h_x(\begin{bmatrix} z \\ \phi(z) \end{bmatrix}) = \frac{\partial h_x}{\partial z} + \frac{\partial h_x}{\partial \phi(z)} \frac{\partial \phi(z)}{\partial z} = J^{h_x}_{0:m} + J^{h_x}_{m:n} \\ \Rightarrow \quad \frac{\partial \phi(z)}{\partial z} = -(J^{h_x}_{m:n})^{-1} J^{h_x}_{0:m} \end{aligned} \tag{15}$$

Nevertheless, its optimality gap varies greatly for different test environments, with the worst case showing as much as a 10% error. Also, the speed of DC3 in testing time is very close to traditional algorithms in nonparallel cases (OPF is always solved sequentially) because of the computation of the physical equations. Its computational time may drastically increase for large grids, raising motivational issues for using neural networks.

3.6 Summary and Research Gap

Table 2 summarizes the literature on solving the OPF problem. Relaxation methods typically do not require ground truth for any learning process, but their outputs are often infeasible, violating important constraints. Supervised learning methods are trained to predict optimal solutions; however, feasibility is not guaranteed. Hybrid learning methods using traditional solvers can ensure feasibility, but they suffer from slow convergence speeds. Conversely, hybrid methods incorporating power flow equations have errors in reconstructing power demand.

Our **Proposal 1** aims to predict feasible and optimal solutions without relying on traditional solvers. Existing unsupervised learning methods in the literature depend on traditional power flow solvers and do not always converge to optimal solutions. In **Proposal 2** and **Proposal 3**, we introduce two novel unsupervised learning mechanisms designed to achieve both feasibility and optimality.

The three papers proposed in this thesis address the research gap and contribute to the field of physics-informed machine learning for optimization problems, providing guarantees on both feasibility and optimality.

Table 2: Comparison of Existing Machine Learning Based-Approaches for Solving OPF Problems

Category	Approach	Existing study	Metrics in consideration		Ground truth	Traditional PF Solvers
			Feasibility	Optimality		
Convex Relaxation	DCOPF	[23–28]	×	×	×	×
	SDP Relaxation	[16, 33, 34]	×	✓	×	×
	SOCP Relaxation	[35–38]	×	×	×	×
IPM	MIPS	[40]	✓	✓	×	✓
Supervised learning	Learn load-optimal solution mapping	[45–49]	×	✓	✓	×
	Emulate traditional solver by steps	[50]	×	✓	✓	×
Hybrid learning	Learn optimal dispatch and voltage magnitude	[51–54]	✓	✓	✓	✓
	Output as warm start point for traditional solver	[56, 57]	✓	✓	✓	✓
	Learn optimal voltage magnitude and phase angle	[59–62]	×	✓	✓	×
	Proposal 1	[67]	✓	✓	✓	×
Unsupervised learning	Penalize the model to learn optimal point	[63–66]	✓	×	×	✓
	Proposal 2	[68]	✓	✓	×	×
	Proposal 3	[69]	✓	✓	×	×

4 Physics-informed Supervised Learning for OPF

4.1 Introduction

Due to the increasing uncertainties brought about by the penetration of renewable energy and the highly stochastic nature of power demands, the OPF problem needs to be solved with finer granularity to ensure system stability and economical operation [70]. However, this problem is proven to be NP-hard [14], making it challenging to solve efficiently. Recently, significant attention has been focused on leveraging state-of-the-art machine learning models to shift the computational burden from online optimization to an offline training process, thus enabling fast real-time OPF decisions. In this chapter, a physics-informed supervised learning framework is proposed, aimed at generating the optimal solution for the ACOPF problem while enhancing its feasibility. This approach entails predicting the optimal voltage magnitudes and phase angles at all buses within the target power system, with the learning goal of minimizing both prediction error and physics-informed power injection reconstruction loss. Subsequently, any feasibility-related errors are systematically eliminated using a novel iterative calibration algorithm. This method utilizes Gauss-Seidel updates with warm-start points on load buses while directly adjusting power injection on generator buses. It converges for all test scenarios on the 14-bus grid and achieves a 92.2% convergence rate on the IEEE 118-bus system. These results surpass all state-of-the-art supervised learning-based ACOPF algorithms.

4.2 Background on Supervised Learning

Supervised learning is a fundamental approach in the field of machine learning, where the goal is to learn a mapping from inputs to outputs based on a collection of data points. There are four components in a supervised learning framework: datasets, machine learning models,

loss function, and optimization algorithms.

- **Dataset:** The dataset is the foundation of supervised learning. The problem needs to be framed into an input-output paradigm, where inputs are the attributes and outputs are the target values of interest. A labeled dataset needs to be collected beforehand, and it is commonly split into training, validation, and testing sets. In supervised learning, the dataset is crucial because it provides the necessary information for the model to learn. The training set is used to train the model, the validation set is used to tune hyperparameters and prevent overfitting, and the testing set is used to evaluate the model's performance on unseen data.

- **Models:** Models are mathematical representations that map inputs to outputs. Various types of models can be used in supervised learning. For example, linear regression, decision trees, support vector regression, and neural networks can be used for regression problems. These models can be seen as functions in which the parameters are optimized to output the expected result given the input.

- **Loss function:** The loss function is a crucial part of supervised learning. In this framework, the learning (optimization) of the model is driven by errors between predictions and actual target values. The loss function quantifies this difference and guides the optimization process during the training of machine learning models. As such, the model aims at decreasing the value of the loss function, thereby improving its predictions. The commonly used loss function for regression problems is Mean Square Error (MSE) [71], defined as follows:

$$L_{mse} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{16}$$

where N is the total number of training points, y_i is the target output, and $\hat{y}_i$ is the

predicted output produced by the machine learning model.

- **Optimization algorithms:** Optimization algorithms are methods used to adjust the model's parameters to minimize the loss function. The basic first-order algorithm is gradient descent, including batch gradient descent, which calculates the loss function and its gradient for the whole dataset in each iteration; stochastic gradient descent, which calculates them for a single point each time; and mini-batch gradient descent, which updates via a group of training points. To overcome local optima, saddle points, and slow convergence, more advanced algorithms with momentum and adaptive actions have been proposed, such as the Nesterov accelerated gradient algorithm, RMSprop, and Adam algorithm [72].

4.3 The Proposed Algorithm

As discussed in Section 3.3 and 3.4, data-driven models in the literature commonly treat ACOPF as a black-box function, taking input as the power demand and generating optimal solutions as output. However, it is vital that these solutions adhere to all the physical engineering constraints **[C1]**-**[C6]** in Definition 3. Previous approaches, such as those proposed in [51–54], separate the ACOPF variables into independent variables predicted by deep neural networks (DNNs) and dependent variables calculated through conventional iterative power flow solvers, such as the GS algorithm [18]. Nevertheless, these solvers experience slow convergence, e.g., only accelerating the ACOPF by less than one order of magnitude [61], especially when applied to large power grids. Hence, generating complete ACOPF solutions without traditional solvers is preferred. Another categorizing method [59–62] aims to produce optimal voltages from DNN models, but the side effect is the unavoidable errors on power demand at load buses. These factors motivate the proposal of a hybrid supervised learning model in this chapter. The proposed method consists of two stages. In the first

stage, the proposed supervised learning model not only considers the prediction accuracy of optimal voltages but also ensures that the input to the system is recovered based on the output. To further guarantee adherence to all constraints, a feasibility calibration method is proposed in the second stage to eliminate any imbalance in the output solutions. This algorithm leverages the physical parameters of the target power system, iteratively updating local variables at each bus until final convergence is achieved.

4.3.1 Physics-informed Supervised Learning Model

In the first stage of the proposed method, the OPF problem is reformulated into a supervised learning framework. In this dataset, the inputs to the problem—active power demand and reactive power demand—and the outputs—optimal voltage magnitudes and phase angles at each bus, as well as optimal power generation at each generator—are generated by traditional ACOPF solvers and collected as input-output pairs. Next, the output variables are divided into independent and dependent variables. The independent variables include the optimal voltage magnitudes $|V_i^{opt}|$ and phase angles ϕ_i^{opt} at each bus, while the dependent variables consist of the active power generation P_{G_g} and reactive power generation Q_{G_g} at each generator.

The proposed DNN model depicted in Figure 7. The input $x : \{P_{D_i}, Q_{D_i}, i \in \mathcal{N}\}$, encompassing active and reactive power demand for all buses, undergoes processing by a DNN. The output is the prediction of optimal voltage magnitudes and phase angles, i.e., $\{|\hat{V}_i^{opt}|, \hat{\phi}_i^{opt}, i \in \mathcal{N}\}$, which is concatenated with the prediction of optimal active and reactive power dispatch on each generator bus calculated by power flow equations defined in Equations 17 and 18, to form the full solution $\hat{y} : \{|\hat{V}_i^{opt}|, \hat{\phi}_i^{opt}, \hat{P}_{G_g}^{opt}, \hat{Q}_{G_g}^{opt}, i \in \mathcal{N}, g \in \mathcal{G}\}$ of ACOPF.

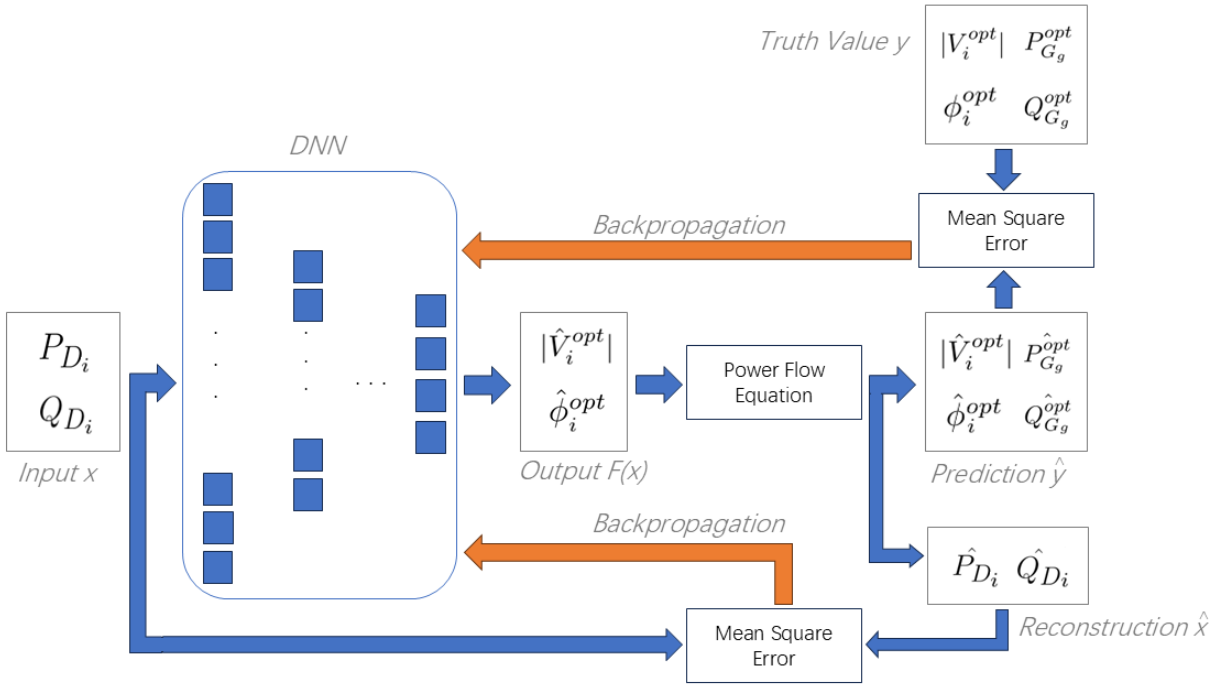

Figure 7: Diagram of End-to-end Learning for ACOPF.

$$\hat{P}_{G_g}^{opt} = P_{D_g} + Re(\hat{V}_g^{opt} \sum_{j \in \mathcal{N}} \hat{V}_j^{opt^*} Y_{gj}^*) \tag{17}$$

$$\hat{Q}_{G_g}^{opt} = Q_{D_g} + Im(\hat{V}_g^{opt} \sum_{j \in \mathcal{N}} \hat{V}_j^{opt^*} Y_{gj}^*) \tag{18}$$

Meanwhile, the inputs of the ACOPF problem, i.e., active and reactive power demand on all buses, are reconstructed as $\hat{x} : \{\hat{P}_{D_i}, \hat{Q}_{D_i}, i \in \mathcal{N}\}$ by Equations 19 and 20. It is notable that there are no reconstruction error on generator buses, because imbalances on these buses are absorbed by the calculation of power generations.

$$\hat{P}_{D_i} = \hat{P}_{G_i}^{opt} - Re(\hat{V}_i^{opt} \sum_{j \in \mathcal{N}} \hat{V}_j^{opt^*} Y_{ij}^*) \tag{19}$$

$$\hat{Q}_{D_i} = \hat{Q}_{G_i}^{opt} - Im(\hat{V}_i^{opt} \sum_{j \in \mathcal{N}} \hat{V}_j^{opt^*} Y_{ij}^*) \tag{20}$$

The proposed mechanism is trained to ensure that the prediction outputs $\hat{y} : \{|\hat{V}_i^{opt}|, \hat{\phi}_i^{opt}, \hat{P}_{G_g}^{opt}, \hat{Q}_{G_g}^{opt}, i \in \mathcal{N}, g \in \mathcal{G}\}$ closely approximate the outputs from commercial ACOPF solvers $y : \{|V_i^{opt}|, \phi_i^{opt}, P_{G_g}^{opt}, Q_{G_g}^{opt}, i \in \mathcal{N}, g \in \mathcal{G}\}$. Additionally, it minimizes the error between the input variables $x : \{P_{D_i}, Q_{D_i}, i \in \mathcal{N}\}$ and the reconstructed inputs $\hat{x} : \{\hat{P}_{D_i}, \hat{Q}_{D_i}, i \in \mathcal{N}\}$, thus establishing a consistent mapping. This consistent mapping enhances an accurate transformation from the input to the output and back to the input, maintaining coherence throughout the process.

The error between $\{x, y\}$ and $\{\hat{x}, \hat{y}\}$ is calculated via mean square error loss, defined in Equation 21, where α and β are two weights for prediction cost on the voltages and power generations. In this work, these two hyperparameters are set to 1.

$$\begin{aligned} L_{mse} =& \mathbb{E}_{i\in\mathcal{N}}\{[(\hat{P}_{D_i} - P_{D_i})^2 + (\hat{Q}_{D_i} - Q_{D_i})^2] \\ &+ \alpha[(|\hat{V}_i^{opt}| - |V_i^{opt}|)^2 + (\hat{\phi}_i^{opt} - \phi_i^{opt})^2]\} \\ &+ \beta\mathbb{E}_{g\in\mathcal{G}}[(\hat{P}_{G_g}^{opt} - P_{G_g}^{opt})^2 + (\hat{Q}_{G_g}^{opt} - Q_{G_g}^{opt})^2] \end{aligned} \tag{21}$$

Then, the learnable weights in the DNN will be updated using the back-propagation algorithm, as described by [73]. Back-propagation is a supervised learning technique that involves calculating the gradient of the loss function L_{mse} with respect to each weight in the model. This gradient represents how much the loss function would change with a small change in the weight. By computing these gradients, the algorithm can determine the direction in which each weight should be adjusted to reduce the loss.

4.3.2 Feasibility Restoration

While physics-informed deep learning has shown superior performance compared to purely data-driven approaches [45, 52, 53], the output of DNNs inevitably contains errors. Conse-

quently, in the context of the ACOPF problem, these errors cause the input parameters x and decision variables $\hat{y}$ to deviate from strict adherence to the equality constraint **[C3]** in Definition 3. For example, it claims that a 1% imbalance on load buses is acceptable and can only be eliminated by controllable distributed energy sources in [61]. However, all errors from the OPF must be eliminated to ensure accuracy and feasibility. In this section, a feasibility calibration algorithm designed to adjust the output from the DNN model is proposed. This algorithm aims to enhance adherence to the power flow equations while avoiding violations of other constraints.

For each load bus $i \in \mathcal{L}$, the complex voltage from the DNN can be updated based on the rule defined in Equation 22. This is the rule used in GS algorithm [18], derived from **[C3]**, leveraging the slack of voltage value on each load bus for a local adjustment based on the current state of the load bus itself and its neighbours.

$$\hat{V}_{i,new}^{opt} = \frac{1}{Y_{ii}}\Big(\frac{-P_{D_i} + jQ_{D_i}}{\hat{V}_{i,old}^{opt}} - \sum_{j \in \mathcal{N}, i \neq j} \hat{V}_j^{opt} Y_{ij}\Big) \tag{22}$$

The power injection at each generator bus $g \in \mathcal{G}$ is updated to eliminate all imbalances, as defined in Equations 17 and 18.

The proposed algorithm operates in epochs. Within each epoch, voltages on all load buses are updated at first, followed by adjustments to power injections at generator buses. The algorithm exits either when a predefined stop criterion ρ for average power flow imbalance at each bus is met or when the maximum epoch E is reached. The algorithmic framework is summarized in Algorithm 1. The *Clip* function is defined in Equation 23, where the target variable is z, whose upper and lower bounds are $\overline{z}$ and $\underline{z}$ respectively. The *Check* function returns residuals regarding the constraint **[C3]**.

$$Clip(z) = \max(\min(z, \overline{z}), \underline{z}) \tag{23}$$

This algorithm has three major advantages compared to the GS algorithm. Firstly, the initial values of the calibration process based on the results from the supervised learning model, incorporating complete variables in an ACOPF solution that are already very close to both feasibility and optimality. In contrast, GS algorithm only utilizes two variables at each bus, with the others starting from random guesses. Secondly, the proposed method updates load buses first and the subsequent update on generator buses, allowing the imbalances from the updates of voltages on load buses to be absorbed by the generator buses. Finally, instead of fixing the active power injection at generator buses, we adjust both active and reactive power injections to utilize the slack on generators, leading to faster convergence.

Algorithm 1 Calibration of solution for ACOPF.

function CORRECTION($x, \hat{y}, E, \rho$)
  $e \leftarrow 0$
  r = $Check$ [C3]
  while $|r| > \rho$ **and** $e < E$ **do**
    $e \leftarrow e + 1$
    for each $i \in \mathcal{L}$ **do**
      $\hat{V}^{opt}_{i,new} = \frac{1}{Y_{ii}}\left(\frac{-P_{D_i}+jQ_{D_i}}{\hat{V}^{opt}_{i,old}} - \sum_{j\in\mathcal{N},i\neq j} \hat{V}^{opt}_{j} Y_{ij}\right)$
      $Clip(\hat{V}^{opt}_{i,new})$
    end for
    for each g $\in \mathcal{G}$ **do**
      $\hat{P}^{opt}_{G_g} = P_{D_g} + Re(\hat{V}^{opt}_{g} \sum_{j\in\mathcal{N}} \hat{V}^{opt^*}_{j} Y^*_{gj})$

      $\hat{Q}^{opt}_{G_g} = Q_{D_g} + Im(\hat{V}^{opt}_{g} \sum_{j\in\mathcal{N}} \hat{V}^{opt^*}_{j} Y^*_{gj})$
      Clip($\hat{P}^{opt}_{G_g}$), Clip($\hat{Q}^{opt}_{G_g}$)
    end for
    r = $Check$ [C3]
  end while
end function

4.4 Experiments

This section will first detail the environment used in this research, including the software, libraries, computation platform, and other relevant tools. Following this, the datasets used for training, validation, and testing are introduced. Subsequently, model selection, hyper-

parameter tuning, and comparative studies are presented. Finally, the performance of the feasibility correction algorithm is illustrated.

4.4.1 Environment and Datasets

In this work, the training datasets were generated using a Matlab script with Matpower 7.0 [40] on a local laptop equipped with an Intel Core i7-10750H CPU and 32.0 GB RAM. The learning algorithm was implemented in Python 3.7 with TensorFlow 2.14.0 and Keras 2.7.0. The training process was conducted on Google Colaboratory, a cloud-based platform offering access to Google's formidable computing resources, including GPUs and TPUs. Specifically, an NVIDIA Tesla V100-SXM2-16GB was used. Chapters 5 and 6 use the same environment settings as this work.

To assess the performance of the proposed algorithm, all experiments were conducted on two standard benchmark systems: IEEE 14-bus and IEEE 118-bus systems. The dataset utilized in this study was generated using MATPOWER's primal-dual interior point solver (MIPS) [40]. Variations in the systems were introduced by randomly varying the nodal active and reactive power demands, ranging from 80% to 120% of their respective nominal values, thereby simulating real-world fluctuations in power demand. This practice is common in the literature, e.g., [45, 52, 61]. For each of these benchmark systems, a dataset comprising 100K pairs of input and optimal output data was generated. This extensive dataset was then divided into three subsets: 80% for training, 10% for validation, and 10% for testing, which is based on the common practice in machine learning.

4.4.2 Performance of the Proposed DNN

The proposed DNN model is selected to be a Convolutional Neural Network (CNN), which uses convolutional filters to extract meaningful patterns and features from input data. Com-

pared to fully-connected neural networks, CNNs are more weight-efficient due to their design of weight sharing and sparse connectivity [73]. To remove the inequality constraints on individual decision variables defined in [C4] and [C5], sigmoid activation functions are used in the output layer of the DNN. The sigmoid function's S-shaped curve, bounded between 0 and 1, allows the model to map the DNN's output into the permissible range specified by the lower and upper bounds of the variables. This mapping is captured by Equation 28, where F_k^L denotes the k-th output from the final layer through a sigmoid function, and x_k^{min} and x_k^{max} are lower and upper bounds for each variable.

$$F_k(x) = F_k^L \cdot (x_k^{max} - x_k^{min}) + x_k^{min} \tag{24}$$

Candidate model settings with different hyperparameters including number of layers, number of filters, activation function, and batch size are listed in Table 3. The model selection is based on their performance on the validation set. The model with the best performance for each testing grid is labeled with †, and selected to be the final model $F(\cdot)$.

Table 3: Different Settings of DNN's Hyperparameters

Power Grid	**Settings**	**Layers**	**Activation**	**Batch**
IEEE 14-BUS	MODEL-1 †	512|256|128|64	LEAKYRELU	512
	MODEL-2	256|128|64|32	RELU	512
	MODEL-3	768|512|128	LEAKYRELU	256
	MODEL-4	64|32	RELU	128
IEEE 118-BUS	MODEL-1	512|256|128|64	LEAKYRELU	256
	MODEL-2 †	512|256|128|64	LEAKYRELU	512
	MODEL-3	768|512|128	LEAKYRELU	256
	MODEL-4	64|32	RELU	128

The model is composed of 189,148 parameters for the IEEE 14-bus system and 309,164 for the IEEE 118-bus system. This architectural choice provides enough parameters that can be fine-tuned to capture the relationship of the underlying data effectively. In the training

of these models, Adam optimizer is adopted, which is the most common choice in training of DNNs [71].

To provide a comparative study of the proposed approach, the same DNN architecture and training process are adopted for DeepOPF-V [61] and Supervised Learning OPF [51] as two benchmark algorithms. Our proposal introduces an additional penalty on power demand reconstruction compared to DeepOPF-V, and the experiment will demonstrate the efficacy of adding this cost term. The second counterpart uses only supervised learning without incorporating physical knowledge, comparative study will highlight the importance of these additional penalties. DeepOPF [52] is not selected for comparison, as it relies on traditional iterative solvers, which are not preferred in this study.

Unlike original approaches that calculate each variable from a separate DNN, a unified approach is adopted, utilizing a single DNN to predict all variables. This streamlined methodology facilitates a direct comparison of methods with similar computational complexity, thereby enabling a fair assessment of performance.

The detailed results of the comparative evaluation conducted on the IEEE 14-bus and IEEE 118-bus systems are presented in Table 4. For the bus-14 system, all algorithms achieved prediction errors of approximately $1e^{-4}$ for each variable, but the proposed calibration algorithm greatly improves the feasibility regarding the power flow equation defined in **[C3]**. When considering the optimality gap, the proposed method outperforms the others. Moving to the IEEE 118-bus system, DeepOPF-V [61] tends to overfit voltages, but performs inaccurately when using these voltages to recover power injections. Supervised OPF [51] encounters challenges in accurately predicting all variables while ensuring feasibility. However, the proposed algorithm demonstrates smaller prediction errors, lower feasibility violations, and a smaller optimality gap. Therefore, the proposed framework is proved to be a more effective solution, surpassing both counterparts across various metrics.

Table 4: Comparison of DNN's Performance in Different Algorithms

Power Grid	Architecture		Performance						
	Model Type	***#Parameters***	$\|V\|_{mse}$	ϕ_{mse}	$P_{G_{mse}}$	$Q_{G_{mse}}$	***Feasibility***	***Optimality Gap***	***Time (s)***
14	Supervised Learning [51]	190,168	$5.2e^{-4}$	$3.2e^{-4}$	$3.5e^{-4}$	$7.2e^{-5}$	$8.5e^{-3}$	2.25%	0.009
	DeepOPF-V [61]	189,148	$6.8e^{-6}$	$6.2e^{-6}$	$8.3e^{-5}$	$1.9e^{-5}$	$3.5e^{-3}$	0.72%	0.023
	Proposed	189,148	$1.0e^{-4}$	$1.2e^{-4}$	$7.1e^{-5}$	$3.8e^{-5}$	$3.8e^{-3}$	0.58%	0.022
	Proposed†	189,148	$1.2e^{-4}$	$1.4e^{-4}$	$7.3e^{-5}$	$3.7e^{-5}$	$< 1e^{-6}$	0.67%	0.117
118	Supervised Learning [51]	324,504	$3.5e^{-4}$	$1.4e^{-3}$	$5.3e^{-4}$	$5.5e^{-4}$	0.032	0.17%	0.018
	DeepOPF-V [61]	309,164	$1.3e^{-4}$	$2.3e^{-4}$	0.042	$3.7e^{-3}$	0.039	1.03%	0.026
	Proposed	309,164	$3.6e^{-4}$	$4.8e^{-4}$	$3.3e^{-3}$	$4.7e^{-4}$	0.012	0.25%	0.028
	Proposed†	309,164	$3.8e^{-4}$	$5.8e^{-4}$	$2.82e^{-3}$	$3.1e^{-4}$	$< 1e^{-6}$	0.15%	0.173

4.4.3 Performance of the Calibration Algorithm

In Algorithm 3, the early stopping criterion ρ is defined as $1e^{-6}$, ensuring that the calibration process terminates when the change in the objective function becomes negligible. Additionally, the maximum number of epochs for calibration E is set to be 100, providing a balance between computational efficiency and convergence.

The calibration algorithm is evaluated using 10,000 testing points for both the IEEE bus-14 and IEEE bus-118 systems. It achieves a convergence rate of 100% for the IEEE 14-bus system and 92.2% for the IEEE 118-bus system, demonstrating the effectiveness of this approach across diverse scenarios. To visualize the convergence behavior, the average performance curve for the convergence cases is depicted with a solid line in Figures 8 and 9. The plot illustrates how imbalance in the grid is eliminated across epochs. Additionally, the plot includes a shaded interval between the best and worst cases for each epoch, demonstrating the statistical patterns of the convergence algorithm across different scenarios. Convergence with average imbalance below ρ is typically achieved around the 33rd epoch for the IEEE 14-bus system and the 40th epoch for the IEEE 118-bus system.

Furthermore, the optimality of the calibrated ACOPF solutions is assessed to evaluate the degree of changes introduced by the calibration process. The proposed algorithm exhibits minimal calibrations, with the average optimality gap experiencing only slight changes. For

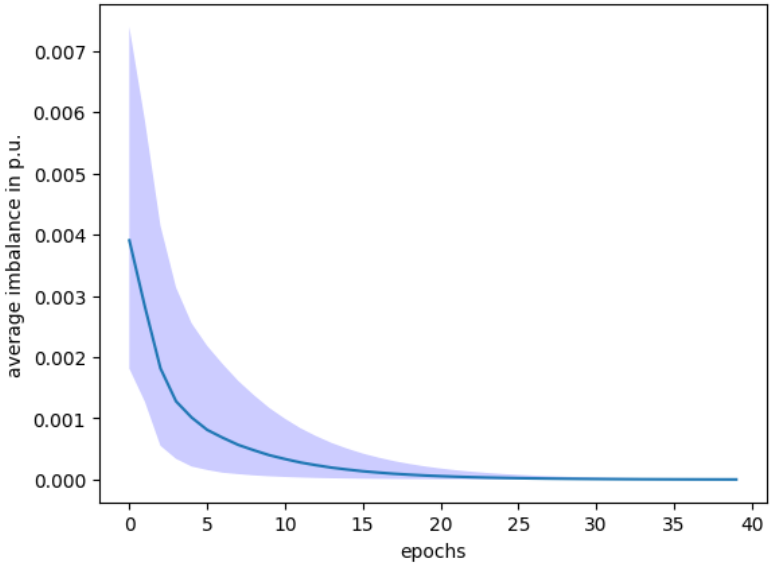

Figure 8: Average Curve of Calibrations on IEEE Bus-14.

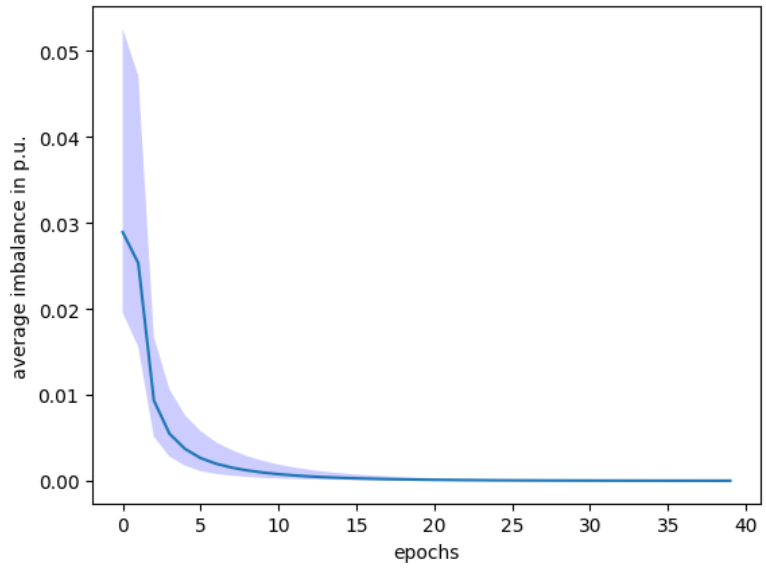

Figure 9: Average Curve of Calibrations on IEEE Bus-118.

instance, the average optimality gap increases from 0.58% to 0.67% for the IEEE 14-bus system and decreases from 0.25% to 0.15% for the IEEE 118-bus system. These negligible changes demonstrate the ability of the proposed algorithm to refine ACOPF solutions while preserving their optimality.

4.5 Summary

In summary, this chapter proposes a physics-informed DNN framework for addressing the ACOPF problem in power grid operations. By leveraging DNNs and incorporating physics-informed penalties, the complex relationships between power demand, optimal voltage magnitudes, and optimal phase angles are captured, while critical constraints and operational requirements are adhered to. Moreover, the proposed calibration algorithm has showcased 100% convergence rates on the IEEE 14-bus system and 92.2% on the IEEE 118-bus system with negligible perturbations to ACOPF solutions from the DNN, thus demonstrating its practical utility in real-world scenarios.

5 Unsupervised Generative Model for OPF

This chapter introduces a novel unsupervised learning model for the OPF problem. The proposed method removes the need for labeled data from traditional OPF solvers, relying instead on the abundant historical operational data available in ISO or utility company datasets. The experimental results demonstrate the efficacy and efficiency of this approach, underscoring its potential for solving OPF using unsupervised learning techniques.

5.1 Introduction

Supervised learning method and hybrid learning algorithms requires the procedure of generating labelled datasets. in the context of OPF, it often runs traditional OPF solvers to obtain input power demand and corresponding optimal solutions. While these methods can achieve high accuracy, they are limited by the availability and quality of labeled data. In response to these limitations, we leverage the abundant historical operational data stored in ISO or utility's data centres and propose an unsupervised learning method to learn the underlying data distribution and search for the optimum. The unsupervised learning model is a novel extension of generative machine learning that effectively addresses the OPF problem in the power grid in real-time. Due to the lack of controllability of the latent space and the more blurred data quality in Variational Autoencoders (VAEs) [74], as well as the slow prediction speed of Diffusion Models [75], a Generative Adversarial Network (GAN) [76] is selected as the machine learning model for this study.

The proposed model is an information-theoretically guided continuous conditional GAN that learns the feasible data distribution and simultaneously extracts a disentangled latent variable for controlling the power generation cost in a completely unsupervised manner. This approach allows the model to generate feasible power flow solutions directly from the input

data, without collecting optimal solutions for training. To ensure the generated solutions are both feasible and optimal, the Karush-Kuhn-Tucker (KKT) conditions are established. The KKT conditions are necessary for optimality in non-linear programming problems and help guide the GAN to produce solutions that meet both the physical constraints of the power system and the optimality criteria.

5.2 Background on Generative Adversarial Networks

Generative Adversarial Networks (GANs) [76] is a framework using a minimax two-player game to train a generative model G and a discriminative model D. Model G captures the data distribution of the training dataset, while model D estimates the probability that an input data point came from the training data rather than synthetic data generated by model G. In the literature, model G and model D are also called the generator and the discriminator, respectively. However, they are consistently refered as models G and D in this chapter to avoid potential confusion because there are also power generators in power grids. The input of model G is a set of random noise called latent variables denoted by z, and the goal of model G is to learn the mappings from the latent space to the data space where the target dataset lies. Model D has a probability of 50% to receive fake data from the output of model G and real data from the dataset. It is trained to maximize the probability of correctly classifying both real and fake data. Mathematically, the minimax game is given by the following loss function in Equation 25.

$$\begin{aligned}\min_G \max_D V(D,G) = &\mathop{\mathbb{E}}_{y\sim P_{data}} [\log D(y)] \\ &+ \mathop{\mathbb{E}}_{z\sim P_{noise}} [\log(1 - D(G(z)))]\end{aligned} \tag{25}$$

Essentially, the loss function of GAN quantifies the similarity between the generative data distribution and the real sample distribution by JS-divergence [77] when model D is optimal. In the typical setting, model D is only used during the training of GAN, but not involved in the prediction stage. Notably, the latent variable z has no meaning other than its meaning via the generative model.

Conditional GANs (cGANs) [78] and its continuous version [79, 80] are extensions of the standard GANs where both model G and model D are conditioned. This condition can be either discrete or continuous, e.g., class labels, text descriptions, or any other auxiliary information. The objective function for cGANs is shown in Equation 26:

$$\begin{aligned} \min_G \max_D V(D,G) = & \mathop{\mathbb{E}}_{x,y\sim P_{data}} [\log D(y|x)] \\ & + \mathop{\mathbb{E}}_{z\sim P_{noise}, x\sim P_{data}} [\log(1 - D(G(z|x)))] \end{aligned} \tag{26}$$

where x represents conditions, which gives the model more control over the generated data based on the context.

Clearly, $G(z|x)$ will be different while manipulating the value of the condition x. Similarly, continuous cGAN [80] extends cGAN by adding a weighted regularization term to the cost function that is the norm of the Jacobian matrix of model G with respect to the condition x, formulated in Equation 27.

$$\begin{aligned} \min_G \max_D V(D,G) = & \mathop{\mathbb{E}}_{x,y\sim P_{data}} [\log D(y|x)] \\ & + \mathop{\mathbb{E}}_{z\sim P_{noise}, x\sim P_{data}} [\log(1 - D(G(z|x)))] \\ & + \lambda \mathop{\mathbb{E}}_{x,z} \nabla_x ||G(z|x)|| \end{aligned} \tag{27}$$

Because there exist infinite number of possible conditions, using finite samples to train the model still leaves gaps in the conditional space. Minimizing the norm of the Jacobian matrix

encourages model G not to perform sensitively to the small changes in the vicinity of the known training conditions to fill these gaps.

5.3 The Proposed Model

The architecture of the proposed model is illustrated in Figure 10, comprising three neural networks: model G, model D, and model Q. This model can be analyzed in two parts: the conditional GAN (cGAN), which includes model D and model G, and the representation learning component, which is achieved through the integration of model Q. Additionally, a search stage based on KKT conditions is introduced. Notably, model D outputs a feasibility score, which is used to select candidate points during the search stage.

5.3.1 Conditional GAN in the Proposal

Model G takes in as input the latent variables c and z (both are random variables), active power demand P_{D_i}, and reactive power demand Q_{D_i}. The active and reactive power demand together are depicted as x and $\hat{x}$ for fake and real conditions, respectively. The input x to model G can be either real conditions drawn from the dataset, or fake conditions sampled in the same distribution of $\hat{x}$.

OPF solutions $G((c, z)|x)$ from Model G and $\hat{y}$ from the real dataset contain active power supply P_{G_g}, reactive power supply Q_{G_g}, voltage magnitude $|V_i|$, and phase angle ϕ_i at each bus $i \in \mathcal{N}$. This setting trains the proposed cGAN to generate random feasible solutions regarding the input condition x.

To accelerate the learning process of generating solutions and enhance the feasibility of the output, we add two types of inductive biases onto model G to shrink the searching space. Firstly, the output layer of model G is modified according to Equation 28, in which

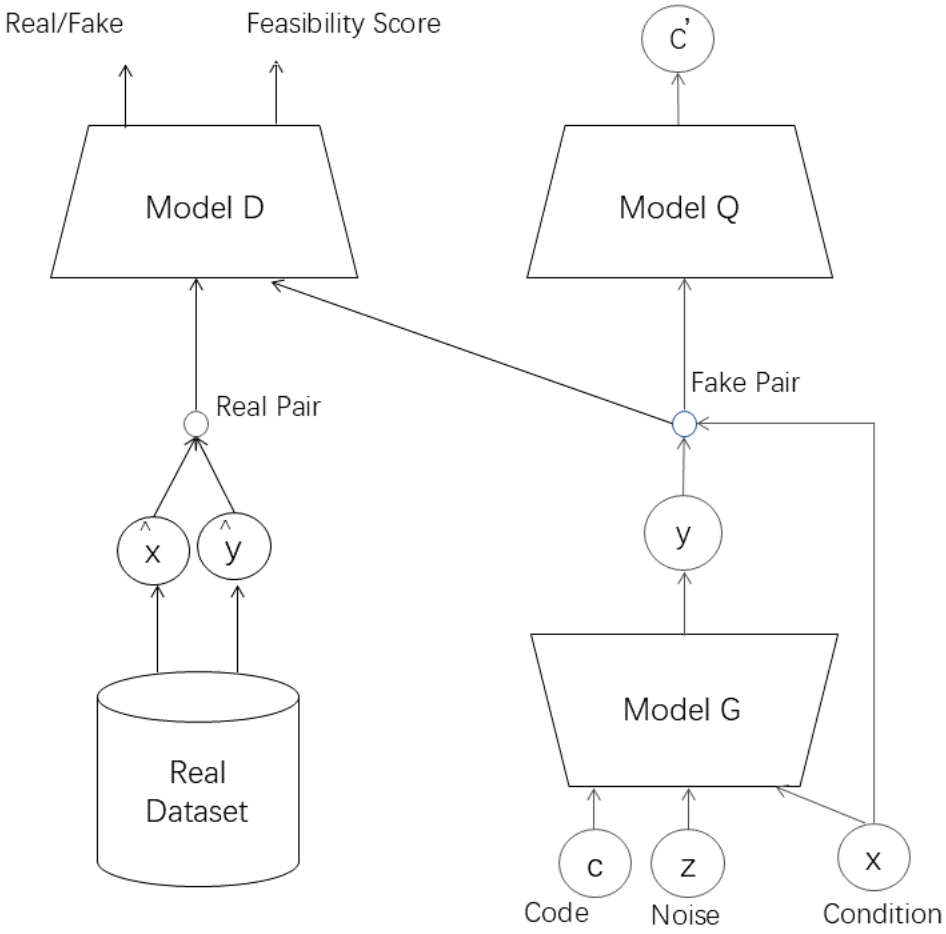

Figure 10: Architecture of the Proposed GAN Model

the original output layer f_L uses the Sigmoid activation function ranging from 0 to 1. This mechanism maps the original output into the range of $[y_i^{min}, y_i^{max}]$

$$f_{out}(f_L(y_i)) = f_L(y_i) \cdot (y_i^{max} - y_i^{min}) + y_i^{min} \tag{28}$$

to satisfy the inequality constraints. Secondly, Equations 8 and 9 are transformed into a physical loss term in Equation 29 to guide model G in satisfying the active and reactive balance constraints.

$$\begin{aligned} L_{Ph} = \sum_{i \in \mathcal{N}} L_{Ph}^i = \sum_{i \in \mathcal{N}} \{ & [(P_{G_i} - P_{D_i}) - \sum_{j \in \mathcal{N}} Re(V_i(V_i^* - V_j^*) y_{ij})]^2 \\ & + [(Q_{G_i} - Q_{D_i}) - \sum_{j \in \mathcal{N}} Im(V_i(V_i^* - V_j^*) y_{ij})]^2 \} \end{aligned} \tag{29}$$

The ACOPF problem can be seen as searching for an optimal solution in a conditional distribution $P(y|x)$, and the condition x is continuous. Thus, the problem naturally motivates the use of continuous cGANs [79, 80]. Model G and D in Figure 10 form the cGAN part, using Equation 27 plus the physical loss in Equation 29 as the cost function. It is designed to understand the feasibility of generated points, including equality and inequality physical constraints. It not only learns under conditions sampled from the condition space during the training process, but also generalizes for unseen conditions. On the other hand, cGAN only generates random feasible solutions conditioned on power demands, but it is still not sufficient for finding the optimum.

There is an additional module in model D within this cGAN model, which modifies the last layer of model D to include a prediction output for the physical imbalance regarding constraint **[C3]**, i.e., the value of Equation 29. As shown in DC3 [65], evaluating the feasibility of solutions by calculating physical violations L_{Ph} is time-consuming in real-time algorithms. Therefore, by leveraging the fast computation capabilities of DNNs, this term

can be accurately predicted via model D, avoiding the computational burden of evaluating physical constraints in large-scale grids during real-time control. The goal of using DNNs is to shift the computational burden from the online (testing phase) to the offline (training phase), enabling fast feasibility measurement during the testing stage. This regression task in model D is defined as $D_R(\cdot)$, and the regression loss is defined in Equation 30:

$$L_R = \mathop{\mathbb{E}}_{x,y} ||L_{Ph} - D_R(x, y)||^2 \tag{30}$$

where L_{Ph} is zero when x and y are from the real dataset, and the prediction for physical loss is made when x and y are from model G. This modification changes the role of model D to not only participate in associated training of model G, but also optimum searching in the second stage.

5.3.2 Information Theoretically Guided Representation Learning

The latent variable z in traditional GANs and cGANs represents the real data distribution in a lower-dimensional space [76]. Commonly, z is neither explainable nor controllable because it is not restricted in how model D may use it. This section explores the controllability of latent variables and introduces an information-theoretically guided component into the cGAN, inspiring the model to learn a single dimension of the latent variable that captures the most salient representation in the power flow dataset. By manipulating the value of this dimension, the outputs of model G will change drastically within the valid distribution. The experiment shows that this dimension is highly correlated to the cost of power dispatch and acts as a control knob.

In the proposed model, a single dimension of the latent variable is isolated and assigned a separate name, 'the latent code,' and notation c for convenience in illustrations. Let's

consider the mutual information between two random variables c and $y = G((c, z)|x)$ in Equation 31,

$$I(c; y) = H(c) - H(c|y) \tag{31}$$

where $H(\cdot)$ is the entropy measuring the information obtained about c by observing y. If $H(c|y) = H(c)$, the two variables are independent, and $I(c; y)$ is zero. The maximum $I(c; y)$ is $H(c)$ when $H(c|y) = 0$, in which case having y reveals everything about c, vice versa (mutual information is symmetric). Therefore, an additional task for the cGAN is to maximize the mutual information between the latent code c and the fake data $y = G((c, z)|x)$. This task will enforce the latent code to carry as much information about the synthetic data as possible in a single dimension. In other words, c learns the most salient feature about y. Notably, the second term in Equation 31 is not computable, as the distribution of fake data is unknown. A variational lower bound for the mutual information is derived in [81] as shown in Equation 32,

$$\begin{aligned} I(c; y) &= H(c) - H(c|G((c, z)|x)) \\ &\geq H(c) + \mathbb{E}_{y \sim G((c,z)|x)} \mathbb{E}_{c \sim P(c|y)} \log Q(c|y) \\ &= L_I(c; G((c, z)|x)) \end{aligned} \tag{32}$$

The intuition of the lower bound above is the non-negativity of KL Divergence. By

changing the order of integrals, the lower bound can be in the following form:

$$
\begin{aligned}
&L_I(c; G((c,z)|x)) \\
&= H(c) + \int_{y\sim G((c,z)|x)} P(y) \int_{c\sim\mathcal{C}} P(c|y) \log Q(c|y)\, dcdy \\
&= H(c) + \int_{y\sim G((c,z)|x)} \int_{c\sim\mathcal{C}} P(y,c) \log Q(c|y)\, dcdy \\
&= H(c) + \int_{c\sim\mathcal{C}} \int_{y\sim G((c,z)|x)} P(y,c) \log Q(c|y)\, dcdy \\
&= H(c) + \int_{c\sim\mathcal{C}} P(c) \int_{y\sim G((c,z)|x)} P(y|c) \log Q(c|y)\, dcdy \\
&= H(c) + \mathop{\mathbb{E}}_{c\sim\mathcal{C}} \mathop{\mathbb{E}}_{y\sim G((c,z)|x)} \log Q(c|y)
\end{aligned}
\tag{33}
$$

In this context, $Q(\cdot)$ can be seen as an inference neural network recovering the latent code c from the synthetic data y. Because latent code c is a known random variable sampled from a uniform distribution, $H(c)$ is a fixed term. Therefore, maximizing Equation 33 is equivalent to recovering latent code c from the synthetic data as accurately as possible. This process can be reframed into an Autoencoder model [82]: taking c as input, encoding it to y, and decoding back to c' with an MSE loss function to be optimized. In Figure 10, the architecture of an Autoencoder can be clearly observed on the right-hand side, with model G and Q as encoder and decoder, respectively. Intuitively, the mutual information loss can be rewritten into an MSE loss in the form of Equation 34:

$$
L_{MI} = \mathop{\mathbb{E}}_{c\sim P(c), z\sim P_{noise}} ||Q(G((c,z)|x)) - c||^2 \tag{34}
$$

Wrapping everything together into the cost function of the GAN model in Equation 35,

in which the third term L_{GP} denotes the gradient penalty in Equation 27.

$$\begin{aligned}\min_G \max_D V(D,G) = & \mathop{\mathbb{E}}_{\hat{x},\hat{y}\sim P_{data}} [\log D(\hat{y}|\hat{x})] \\ & + \mathop{\mathbb{E}}_{c\sim P(c), z\sim P_{noise}} [\log(1 - D(G((c,z)|x)))] \\ & + \lambda L_{GP} + \nu L_{MI} + \mu L_{Ph}\end{aligned} \tag{35}$$

Moreover, the implementation of InfoGAN proposed in [81] suggests that model Q shares most of the hidden layers with model D to accelerate the training. However, we expect a separate inference network Q to recover information only from the active power generation because the power generation cost in Definition 3 relates solely to the active power supply P_{G_g}. Other dimensions of the output from model G can be safely discarded in model Q due to their irrelevance to the cost function.

5.3.3 Searching for the Optimal Solution

The proposed ML model is trained using a dataset composed of feasible points that are not necessarily optimal. The model G synthesizes various values of y for fixed inputs x when latent variables are altered. The latent variables are composed of two parts - a noise vector z selected from a Gaussian distribution, as typical for GAN systems [76], and the single-dimensional code c. The noise component z allows for variations in the outputs synthesized by G, while the latent code c enables the generation of solutions associated with wide range of costs, which can be leveraged for optimizations.

The degree of feasibility of generated y given x can be determined by the two-dimensional feasibility score f from model D, which quantifies the imbalance in the power flow equations and the violation of branch flow limits. The code c is a single-dimensional independent variable directly correlated to the cost of output y. This can be used to force G to output y

associated with low costs. The score f is utilized to ascertain the feasibility of the solutions generated. Among the finite set of solutions synthesized by G for various values of c and z, the one with the least cost while being feasible is selected. This solution is then further refined via iterative updates to improve costs while maintaining feasibility. This process is outlined next.

The proposed search algorithm is divided into two parts. In the first part, the search space of the outputs generated by G is narrowed down to a specific region pertaining to low-cost solutions as prescribed by c. A finite number of outputs M are sampled from G that fall within this region. The feasibility of these outputs is ranked from lowest to highest based on the feasibility score f. The output y associated with f indicating the greatest feasibility is selected (i.e., low values of f), and this is denoted as y^*.

In the second part of the search, y^* is iteratively perturbed to move in the direction that minimizes an approximation of the Lagrangian of the OPF problem. This approximation, referred to as $\mathcal{L}_a$, does not require the use of traditional solvers.

$\mathcal{L}_a$ is composed of three components:

- 1. Generation costs $C(y)$, which is the economic cost in the OPF problem.
- 2. Estimation of violations regarding the power flow equations.
- 3. Violations of inequality constraints (i.e., [C3] to [C7] in Definition 3).

The point y^* serves as the initial point in the search for the point that minimizes $\mathcal{P}_{OPF}$, which is then iteratively modified in the direction of the negative gradient of $\mathcal{L}_a$. Computing the gradients of $C(y)$ and box inequality constraints with respect to y is straightforward as these are quadratic and linear relations. However, computing the gradient of the power imbalance component is difficult and necessitates knowledge of the underlying topology and

parameters of the system. To simplify this, the model D is trained to compute the degree of imbalances in the power flow equations, which can be used in $\mathcal{L}_a$ as the estimation of the violations. D_f is a mapping that approximates the degree of mismatches in the power balance equations. Assembling all these components together culminates in $\mathcal{L}_a$ as follows:

$$\mathcal{L}_a(x, y, \alpha, \beta, \gamma) = C(y) + \alpha D_f(x, y) + \beta(y_i - \bar{y}_i) + \gamma(\underline{y}_i - y_i))$$

where α, β, γ are analogous to dual variables and x is the primal variable of the optimization problem $\mathcal{P}_{OPF}$. α takes a value in the real space and β, γ are non-negative as these are associated with inequality constraints. To improve point y^*, an iterative update process is defined as follows:

$$y^*_{k+1} = y^*_k - t\frac{\partial \mathcal{L}_a(y^*_k)}{\partial y}$$

where k is the iteration number, and t is a fixed step size. These updates aim to iteratively adjust y^* until an equilibrium point of $\mathcal{P}_{OPF}$ is reached (i.e., the gradient is zero). As this is a non-convex problem, this equilibrium point may be the global optimum. With stage 1 of the proposed search algorithm, the search space is narrowed to be close to the optimal solution with the aid of c. These iterative searches defined in stage 2 will further narrow this gap, as demonstrated in the next section.

For each update step, the gradient of $\mathcal{L}_a$ is computed. This computation is straightforward for all terms except for $D_f(x, y)$. The Jacobian of $D_f(x, y)$ is computationally expensive, especially since this has to be executed for every iteration. The following zeroth-order gradient estimation [83] is used instead:

$$\frac{\partial D_f(x, y)}{\partial y_i} \approx \frac{D_f(x, y + h * e_i) - D_f(x, y - h * e_i)}{2h}$$

where h is the perturbation applied at point y and e_i is the direction in which the perturbation is applied. h is set to 10^{-7} and e_i is the standard basis vector where the i^{th} component is set to 1 and all remaining components are 0.

The above iterative search algorithm is summarized in Algorithm 2. To simplify the expressions, boldface variables are utilized to represent a set of points. For example, $\mathbf{c}$ represents M values sampled uniformly in the listed range. The range selected for sampling c is set to be $[-1, -\epsilon]$ where ϵ can take values in the range $(0, 1]$. This interval is selected as c is set to take values in the range $[-1, 1]$ and since c is directly correlated with the costs, lower costs will lie in the left-most interval. The algorithm uses the threshold ζ for the feasibility score to filter out infeasible solution instances. The stopping criterion selected for this algorithm is the change in sign of all dimensions of the approximated gradient of the Lagrangian. Intuitively, this criterion indicates that a local extrema has been encountered during the iterative refinement process. As approximations of the Lagrangian are used in this algorithm, searching for the precise location of the extrema will not be a useful endeavor. A second stopping criterion is introduced to prevent the iterative refinement process from continuing indefinitely and this is a limit on the maximum number of refinements. Too many refinements can steer the solution to be too far from the cost-effective/feasible region originally identified by the proposed model. The convergence criterion is set to the gradient of all dimensions reaching zero at the equilibrium point or having changed their signs (slightly overshooting), which is checked at the beginning of each iteration.

Next, the theoretical properties of the equilibrium point y^* are presented. In convex optimization problems, the Karush Kuhn Tucker (KKT) conditions are both necessary and sufficient for optimality. In non-convex optimization problems, these are necessary conditions. In the following, it is shown that the iterative algorithm outlined in Algorithm 2 does approximately satisfy the KKT conditions of $\mathcal{P}_{OPF}$. Then in the next section, empirical

Algorithm 2 Iterative Search for Equilibrium OPF Solution

Input: $x, M, t, \epsilon, G(\cdot), D_f(\cdot), \zeta, k_{max}$

$\mathbf{c} \leftarrow$ Sampling M times in $(-1, -\epsilon)$
$\mathbf{z} \leftarrow$ Sampling M times in $\mathcal{N}(0, 1)$
$\mathbf{x} = G((\mathbf{c}, \mathbf{z})|x)$
$\mathbf{f} = D_f(\mathbf{x}, \mathbf{y})$
$f^* = min(\mathbf{f})$
if $f^* > \zeta$ **then**
    return
end if
$y^* = y[argmin(\mathbf{f})]$
$\alpha, \beta, \gamma : [0, ..., 0]^T$
$k = 0$
$s = \text{sign}(\frac{\partial \mathcal{L}_a}{\partial y})$
while $\frac{\partial \mathcal{L}_a}{\partial y_i} = s_i, \forall i$ and k$\leq k_{max}$ **do**
    $\alpha : \alpha + \mathcal{L}_a / \frac{\partial \mathcal{L}_a}{\partial \alpha}$, $\beta : \beta + \mathcal{L}_a / \frac{\partial \mathcal{L}_a}{\partial \beta}$ and $\gamma : \gamma + \mathcal{L}_a / \frac{\partial \mathcal{L}_a}{\partial \gamma}$
    $\beta = \max(0, \beta)$
    $\gamma = \max(0, \gamma)$
    $y^* = max(min(y^* - t\frac{\partial \mathcal{L}_a}{\partial y^*}, \bar{y}), \underline{y})$
    k++
end while
return y^*

experiments conducted on realistic systems demonstrate that these equilibrium points are indeed very close to the optimal solution (i.e. the optimality gap is small).

Theorem 5.1. *$y^*, \alpha, \beta, \gamma$ computed using Algorithm 2 satisfy the Karush-Kuhn-Tucker (KKT) conditions listed as follows:*

1) Stationarity: $\dfrac{\partial \mathcal{L}_a}{\partial y} = 0$

2) Primal Feasibility: [C1] to [C7] *satisfied for y^**

3) Dual Feasibility: $\beta \geq 0, \gamma \geq 0$

4) Complementary Slackness: $\beta(y_i^* - \bar{y}_i) = 0,\ \gamma(\underline{y}_i - y_i^*) = 0$

Proof. KKT conditions are composed of four components: 1) Stationarity; 2) Primal Feasibility; 3) Dual Feasibility; and 4) Complementary Slackness. Stationarity is approximately achieved as the gradient of the Lagrangian is close to zero based on the termination condition

defined in Algorithm 2. Primal feasibility occurs when constraints [**C1**] to [**C7**] are met. This is achieved with Algorithm 2 as the initial starting point of the search is selected to the one with the highest feasibility score, the inequality constraints are embedded within the last layer of G and the search direction maintains feasibility as this is inherent in the gradient calculations. Dual feasibility ensures that the dual variables associated with the inequality constraints are non-negative. This is enforced by lines 15 and 16 of Algorithm 2. Finally, complementary slackness ensures that when the inequality constraints are inactive, the corresponding dual variable is 0 and when these are active, the dual variable is non-negative. Lines 14-16 of Algorithm 2 along with the last layer of G allow for complementary slackness to be satisfied. □

5.4 Experiments

In this section, the performance of the proposed mechanism is evaluated via practical experimental studies. The experiment setup and data generation process will be discussed in Section 5.4.1. The training and performance of the GAN model will be demonstrated in Section 5.4.2. Finally, a case study is conducted to compare the proposed method and traditional methods in real-time OPF in Section 5.4.3.

5.4.1 Datasets

The experiment in this work is conducted on IEEE 118-bus, IEEE 300-bus, and PEGASE 1354-bus systems. In the 118-bus system, there are 54 traditional generation systems connected. The 300-bus and 1354-bus systems are composed of 69 and 260 generator buses, respectively. The datasets used for training, validation, and testing are generated on MatPower 7.1 [40]. The power demands were sampled within the [80%,120%] range of the default load value on each bus uniformly as the common practice in the literature [45, 52, 63]. The

power factor, human interaction and correlation of weather and time factors are not considered in this data generation process. There are 55,966, 48,836, and 67,000 feasible data points for the three benchmark systems, respectively. In each dataset, 85% is used for training the model, 10% for model selection and hyper-parameter tuning, and the remaining 5% for testing. In the training stage, each generator bus's power supply and voltage magnitude are uniformly sampled within their working range, and MatPower completes feasible solutions by solving power flow equations. For testing, MatPower calculates the optimal solution using SDP relaxation [33] and IPM algorithm [40].

Considering the uncertainties from DERs, we assume that each non-generator bus has either wind turbines or solar panels, and these are aggregations of DERs on the target bus. In the U.S. in 2020, wind and solar power generations account for 8.4% and 2.3%, respectively, so the deployment and parameters of DERs in the experiment are designed to match these percentages. The IEEE 118-bus grid contains 64 load buses, where 24 of the load buses are deployed with wind farms, and 40 are with solar stations. Each wind farm is assumed to have one turbine with an 1800 m^2 swept area. Each solar farm has 10,000 panels (1.67 m^2 for each). This experiment only considers the active power generation for both wind and solar power generations, and they will act as negative power demand. The detail for modelling wind turbines and solar PV are demonstrated in Appendix A.

5.4.2 Performance of the Proposed Model

Hyper-parameters of the Model Table 5 tabulates the detailed parameters and settings of the three neural networks in the proposed model for the IEEE 118-bus system. The three networks G, D, and Q are trained with different time scales. The inputs of model G are the power demand x, latent code c being a one-dimensional variable sampled uniformly in the range [-1,1], and latent variable z, a 200-dimensional random variable from the normal

distribution. Model G and Q train once in the outer loop, while D trains ten times in the inner loop. The regression task shares most of the layers of model D, except the last two layers: a hidden layer with four neurons and an output layer. The reason for a much smaller architecture of Q is that it only recovers one single dimension. Our research study shows that simple structures of model Q induce a more linear relationship between latent code c and cost value. All the hidden layers use convolutional layers and the Leaky ReLU activation functions. The weights λ, μ, and ν in Equation 35 are set to be 10, 10, and 1000.

Table 5: Architecture of the Proposal (118-bus system)

Component	Description	Dimensions / Layers
Model G	Input Layer	y:236, c:1, z:200
	Hidden Layers	256, 128, 64, 32
	Output Layer	x: 344
	Activations	LeakyReLU*4, Sigmoid
Model D	Input Layer	y:236, x:344
	Hidden Layers for D_r	512, 256, 128, 64, 32, 16, 8
	Output of D_r	r:1
	Activations for D_r	LeakyReLU*7, None
	Hidden Layers for D_f	32, 16, 8
	Output of D_f	f:2
	Activations for D_f	LeakyReLU*3, None
Model Q	Input Layer	x:54
	Hidden Layers	4
	Activations	LeakyReLU
	Output	$\hat{c}$:1

Training of the Proposed Model Model G's loss curve is depicted in Figure 11. It suffered a low-quality generation period before the 4,000th iteration. After that, it learns to generate close-to-real synthetic solutions and converged at a low loss value. Figure 12 demonstrates the change of the norm of model G's Jacobian Matrix during the training. The norm value keeps falling down to zero, which means the fine-tuned model G is not sensitive to small perturbations to the input power demand.

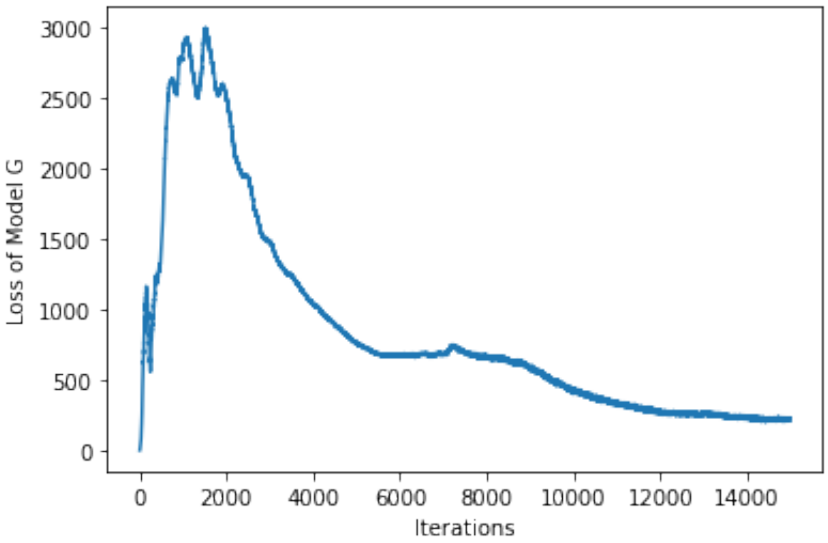

Figure 11: Loss Curves of Model G on IEEE 118-Bus

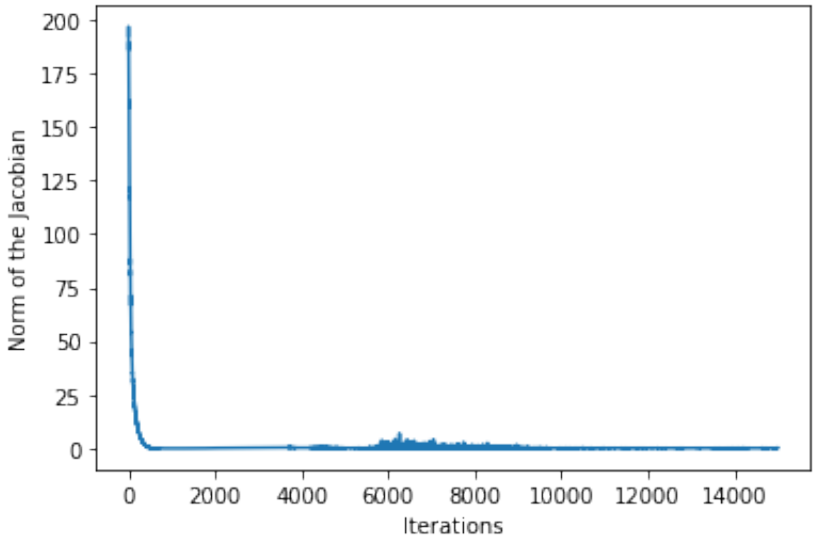

Figure 12: The Curve of the Norm of mModel G's Jacobian Matrix on IEEE 118-Bus

As the learning proceeds, model G learns to generate solutions satisfying power flow equations. The physics equations (Equations 8 and 9) are evaluated every 100 iterations, and the learning procedure is illustrated in Figure 13.

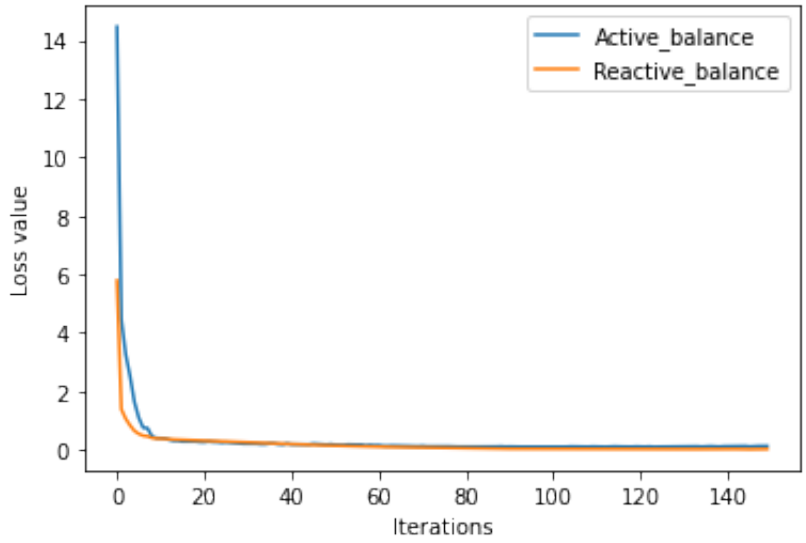

Figure 13: The P/Q Physical Violations of Generated Solutions on IEEE 118-Bus

Model D's loss in learning is shown in Figure 14. The blue curve is the output of model D for real data input, while the orange curve for synthetic data from model G. Model D in the Wasserstein setting should use the sign of its output to determine real (+)/fake (-) data and the absolute value for the confidence. The curve outputs positive for real data and a very negative value for fake data at the early stage of the training. After 40,000 iterations, both curves converge to the same level, close to zero. This means the quality of synthetic data is greatly improved during model G's learning process, and model D cannot even distinguish their differences. Thus, an equilibrium between the two players in this game is somehow reached.

The regression part of model D is simply a minimization task of MSE loss, and it can be either designed inside model D or be a separate network itself. The loss curve falls to zero quickly, as shown in Figure 15.

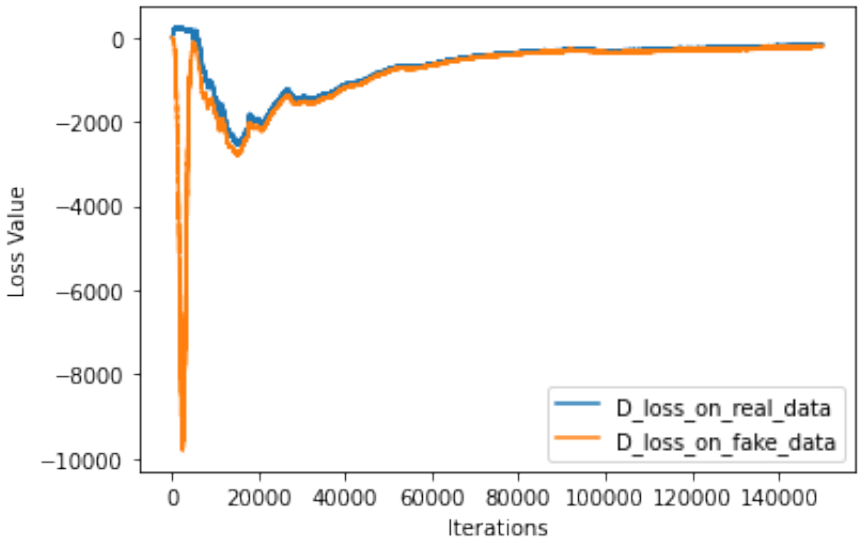

Figure 14: Loss Curves of Model D on IEEE 118-Bus

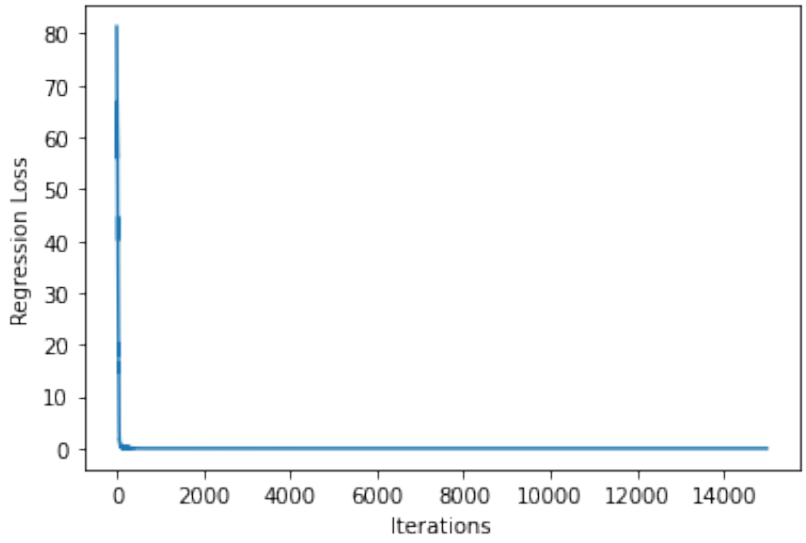

Figure 15: Loss Curves of the Regression Task on IEEE 118-Bus

As discussed in Section 5.2, the sub-system containing model G and Q can be seen as an Autoencoder, which minimizes the reconstruction loss between input c and output of model Q. Figure 16 plots the loss curve of this sub-system. It shows that the Q model successfully recovers the latent code c.

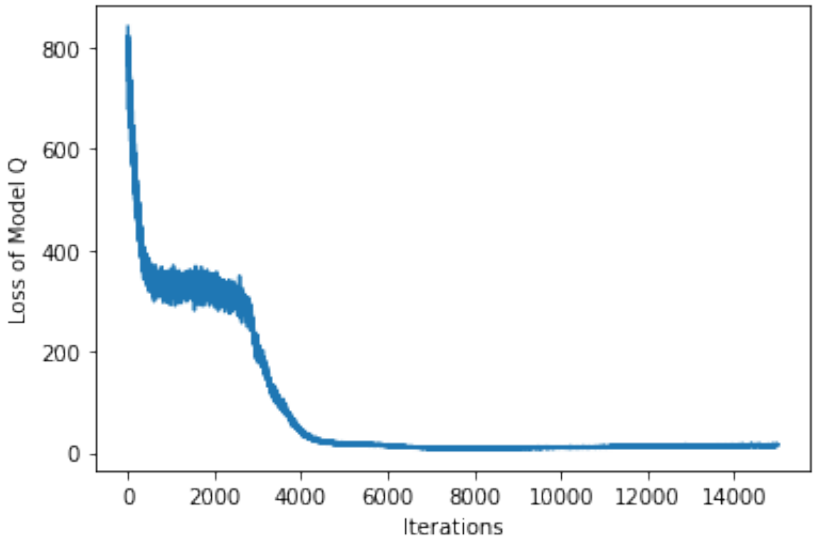

Figure 16: The Loss Curve of Model Q on IEEE 118-Bus

Testing on Latent Code For a given demand x, by fixing the noise z to be a random value and slowly increasing the value of c, the value of the cost function in Definition 3 shows an interesting phenomenon in Figure 17. Its value changes along with different latent code c, and the curves are quite smooth and linear.

The reason for the salient changes corresponding to controlling the value of c is that the existence of model Q forces model G to encode the most crucial aspect of data into c. From the perspective of information theory, many choices of c may exist, but model G picks up the one carrying the most information in the data domain to maximize the mutual information. Indeed, the most significant feature of power generation data is the magnitude of generated active power in the target grid, which directly influences the cost function. It means that

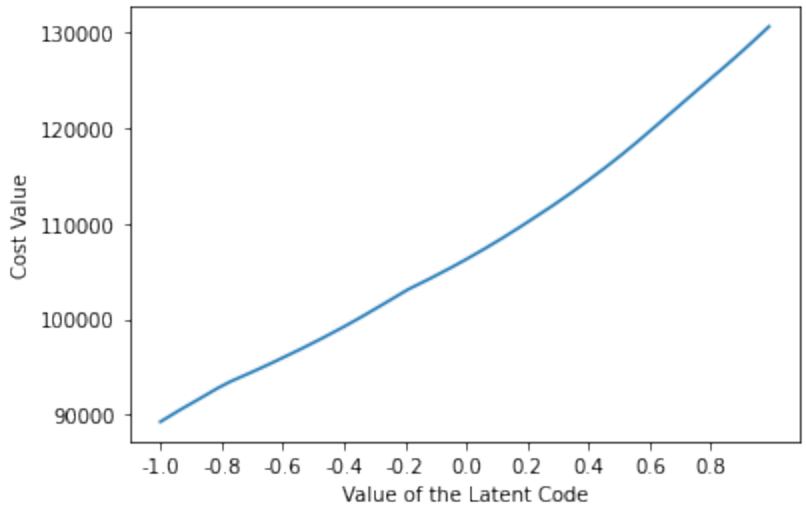

Figure 17: The Relationship Between the Latent Code and the Cost Value on IEEE 118-Bus

by varying the value of c, we can generate feasible solutions with different cost values. Even setting c to be small will minimize the cost function. Besides, it is shown in Table 5 that model Q only contains a single hidden layer, so the effect of its activation functions will be more linear than deeper structures. This is why the simple structure of Q induces close-to-linear behaviors of latent code c. Because the coefficients from MatPower have high values for the first-order term and low values for the second-order term, the cost function also performs linearly with respect to the active power generation.

Optimality Evaluation Next, the optimality gap and time for computation of solutions based on Algorithm 2 is compared with other techniques. For the fairness of the comparison under the same assumptions, traditional solvers SDP [33] and IPM [40] are used as baseline algorithms. The search parameter ϵ in Algorithm 2 is selected to be 0.6. N is varied from 50 to 5000. Table 6 summarizes these results for 1000 different scenarios. The first column lists the three benchmark systems considered in this paper, the second column presents the number of points sampled (i.e. N), the third column lists the average percentage gap and the 95% confidence interval of the solutions computed using our proposed model from the

Table 6: Performance Comparison to SDP and IPM.

Grid	N	Average Gap to SDP(%) +0.95 Confidence Interval	Average Gap to IPM(%) +0.95 Confidence Interval	Time(s)	Feasible
118	50	3.97±0.49	3.79±0.47	0.12	✓
	100	3.87±0.48	3.27±0.39	0.21	✓
	200	3.63±0.45	3.44±0.42	0.27	✓
	500	3.52±0.44	2.97±0.38	0.29	✓
	1000	3.18±0.38	3.00±0.37	0.34	✓
	3000	2.90±0.35	2.70±0.33	0.56	✓
	5000	2.92±0.35	2.71±0.35	0.65	✓
300	50	-	2.01±0.37	0.18	✓
	100	-	1.96±0.26	0.20	✓
	200	-	1.98±0.36	0.24	✓
	500	-	1.98±0.34	0.27	✓
	1000	-	1.92±0.33	0.49	✓
	3000	-	1.94±0.32	0.65	✓
	5000	-	1.96±0.31	1.02	✓
1354	50	-	2.69±0.64	0.32	✓
	100	-	2.67±0.64	0.37	✓
	200	-	2.66±0.68	0.39	✓
	500	-	2.63±0.67	0.47	✓
	1000	-	2.53±0.64	0.61	✓
	3000	-	2.60±0.64	1.03	✓
	5000	-	2.61±0.62	1.58	✓

solutions computed using the SDP solver, the fourth column is the average percentage gap and 95% confidence interval of our proposed model from the solutions computed by the IPM solver, the fifth column presents the average time entailed in computing the solutions using our proposed model and the last column reflects whether or not solutions from the proposed model are feasible based on the outcome of traditional load flow solvers in MatPower.

For all combinations of parameters, it is clear that the optimality gap of the proposed ML model is within 3-4% of the traditional solvers. The 95% confidence intervals indicate small deviations from these average values. Although the gap decreases for increasing values of N,

this is not extensive. Thus, smaller samples can be used as these are associated with lower computational times (i.e. $\leq$ that are lower than 0.5 s). The average computational times entailed by the SDP solver for solving the 1000 scenarios are 98.1 s for 118-bus, 165 s for 300-bus and 241.7 s for 1354-bus systems respectively. For the IPM solver, the computational times are 8.29 s, 8.50 s and 8.76 s for these three systems respectively. Moreover, it is important to note that the converge rates with the SDP solver are between 1.2% and 18.0% for the 300-bus system, and 2.5% and 69.6% for 1354-bus system with the SDP solver. For this reason, the comparisons with our solution to the SDP solver are not included in Table 6. Our solver requires less than 0.5 seconds on average for the two experimental grids. Hence, the improvement in computational speed is at least 16 and 180 times for the 118 and 1354-bus systems respectively in comparison to these traditional solvers. Thus, real-time computations are possible with our proposal. Furthermore, upon evaluating the feasibility of the solutions using external traditional solvers (e.g. load flow solver from MatPower), these verified that every solution obtained from the proposed ML model are indeed feasible for IEEE 118, IEEE 300 bus and Europe 1354 grids. The proposed model produced these solutions without resorting to external solvers and this is not the case with the state-of-the-art data driven solutions like [52, 65].

5.4.3 Case Study on Real-time OPF vs Droop Control

As shown in the experiment, traditional methods such as SDP and IPM take a much longer time for OPF computations, so it is impractical to respond to real-time demand and DERs change. Traditional OPF is conducted in fixed intervals (e.g., every 5 minutes), and its results are used as setpoints for droop control process inside generators. Hence, the actual expense of the power generation at a specific moment is the summation of the cost of the setpoint and additional generation from droop control and AGC. This overall cost can be

formulated in the form of Equation 36:

$$\sum_{g \in \mathcal{G}} C_g(P_{G_g} + \Delta P_{G_g}) \tag{36}$$

where ΔP_{G_g} is the extra power generation from generator $g \in \mathcal{G}$ based on droop control and AGC. This real-time cost cannot be controlled by traditional OPF, and according to [87], additional power generation by droop control and AGC responses at the i-th generator is based on Equation 37.

$$\Delta P_{G_i} = \frac{\Delta P / R_i}{\sum_{g \in \mathbb{G}} 1/R_g} \tag{37}$$

where ΔP is the overall demand increase in the grid, $R_g, g \in \mathcal{G}$ are droop speed regulation related to frequency variation range and megawatt base of the target generator. This case study assumes all generators have the same frequency variation ranges and megawatt bases.

Intuitively in Figure 18, when imbalance happens, the power system takes the extra amount of electricity needed from inertia inside the generators, and then the generators' turbines slow down. AGC will control all connected generators to adjust their fuel levels to restore their nominal frequency.

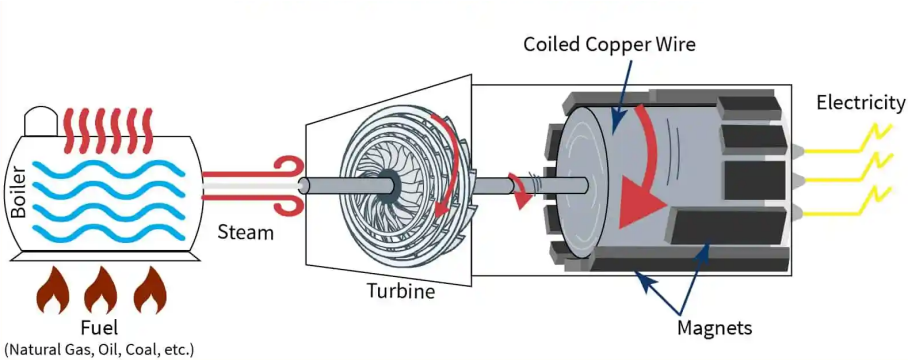

Figure 18: Structure of a Power Generator [86]

The traditional OPF assumes the power demand in the grid remains static or varies in a very small range in the intervals so that the most economical solutions can still be the

optimal or close-to-optimal after droop control. Unfortunately, this may not hold. The OPF computations depend on either the start point of the interval or the prediction at the moment of delivery. Still, any of them may introduce errors. With high uncertainty of load demand and penetration of DERs, the condition may change drastically in the interval, and the literature on load forecasting also reveals unavoidable errors. e.g., Kalman Filter estimator [88] results in about 18% mean absolute percentage error (MAPE), and two deep learning methods [89,90] report 3.9% and 7.4% MAPE with very complex structure (potentially long execution time).

Thus, assuming the load/DERs derivation is +5% in the interval for any reason, this experiment tests the real-time power generation loss of the proposed mechanism and traditional methods with droop control. Droop control will distribute the extra demand to all generators because frequency droop is a global variable when traditional OPF underestimates the load demand. In contrast, the proposed model provides real-time optimal solutions. The result on 1,000 test scenarios shows that traditional OPF + droop control is 8.44% more expensive than our method in +5% load increase cases.

5.5 Summary

In this chapter, a novel information-theoretically guided continuous conditional GAN is proposed that learns the feasible power flow data distribution and draws a controllable latent code to manipulate the economical cost of the generated solutions. By leveraging the latent code along with the establishment of the KKT conditions, the proposed algorithm can successfully generate solutions that are close to globally optimal. This work relaxes the common assumption in the literature that model owners must collect an optimal power flow dataset; instead, it only requires feasible power flow data to train the model. To the best of the author's knowledge, this is the first time a generative model has been used in

a constrained optimization task. This work sheds light on the construction and innovation of generative models and representation learning models in smart grids and constrained optimizations.

6 Data-driven Convex Optimization for OPF

6.1 Introduction

Convex relaxation techniques have been extensively studied to address the ACOPF problem. Methods such as linear relaxation [23–28], SDP relaxation [16, 33, 34] and SOCP relaxation [35–38] have been proposed to relax the original problem into a convex problem that can be solved more efficiently. However, these convex relaxation methods often result in solutions that may violate the original problem constraints, particularly in the case of meshed networks, and can be computationally expensive for large systems due to their formulations.

This chapter proposes a data-driven approach that leverages the input convex neural network (ICNN) model [91] to reformulate the ACOPF problem into a convex problem, thus facilitating its solution in a tractable manner. The ICNN model is designed to learn a convex boundary for the feasible points of the ACOPF problem, transforming the non-convex constraints into convex ones while preserving the problem's original complexity. The proposed method utilizes adversarial training, relying solely on feasible data points, which allows for the use of existing grid datasets for training purposes. Given that the objective function of the original ACOPF problem is typically a convex quadratic function [22], the reformulated problem remains convex, enabling the computation of globally optimal solutions. Thereafter, a two-stage iterative computationally efficient algorithm is introduced to compute the solution to the reformulated problem that is guaranteed to be electrically feasible. These guarantees are established via theoretical constructs from fixed point iteration and a global convergence condition. A comprehensive studies in five practical benchmark systems along with comparative studies with existing work are presented to demonstrate the effectiveness of this proposal.

6.2 Background on ICNN

ICNN is a specially designed DNN model to handle convex mappings between input and output pairs of data. ICNNs are distinct from traditional DNN models such as feedforward neural networks [73], due to their architectural constraints that ensure the convexity of the output with respect to the input. This property makes ICNN particularly useful in optimization tasks, where convexity is a desirable attribute for ensuring global optimality and convergence. The general architecture of an ICNN is depicted in Figure 19. There are two paths in the ICNN where the first is a feedforward path denoted by W_k^z and shortcut paths W_k^I.

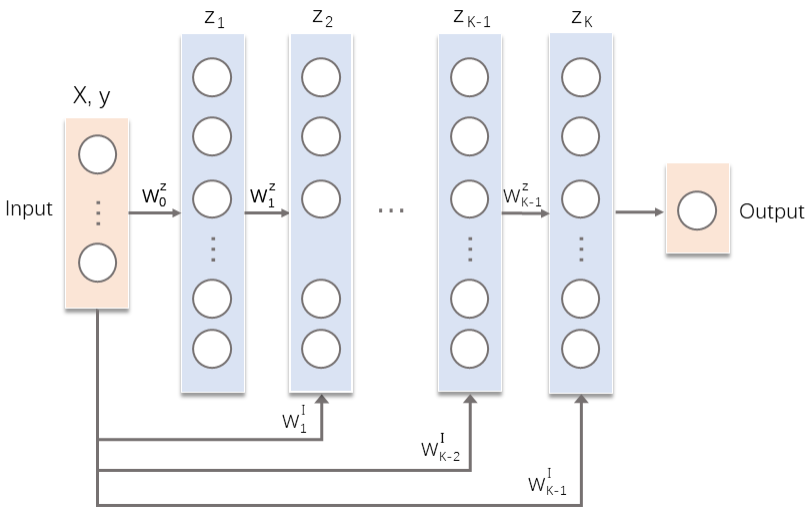

Figure 19: Architecture of ICNN [91].

The output of the k^{th} layer of the ICNN is listed in Equation.38, where the weights W_k^z are weights for the feedforward path in each layer k, W_k^I are the weights for the residual link of input variable (x, y), and b_k are the bias at each layer.

$$Z_{k+1} = \delta_k(W_k^z Z_k + W_k^I(x, y) + b_k) \tag{38}$$

This model's design ensures that the output Z_K is convex with respect to the input x. This is achieved through two key principles:

- **Non-Negative Weights:** The weights in the feedforward path (W_k^z) are constrained to be non-negative.
- **Non-decreasing Activation Functions:** The activation functions used in ICNNs are convex and non-decreasing. Common choices include ReLU (Rectified Linear Unit) and linear activation functions.

The convexity of Z_K with respect to x and y is guaranteed by design as the non-negative sum of convex functions remains convex and the composition of a convex function into a convex non-decreasing function is also convex [91].

6.3 Proposed Method

This section will first explain the rationale and mechanisms for relaxing the OPF problem into a convex optimization problem. It will then introduce the training process of the proposed model. Following that, a searching algorithm and a feasibility correction algorithm will be presented.

6.3.1 Elimination of Non-convexity in ACOPF

Rationale for Convex Approximation of the Feasible Set In Definition 3, **[C3]** and **[C6]** are two groups of constraints that make the ACOPF problem nonlinear and non-convex. The gradient of power injection at bus i with respect to bus voltages, $\frac{dS_i}{dV}$, is defined as follows

in Equation 39.

$$g^i = \frac{\mathrm{d}S_i}{\mathrm{d}V} = \begin{cases} V_i Y_{ii}^* + \sum_{j=1}^{N} V_j^* Y_{ij}^*, & \text{if } i = j \\ V_i Y_{ij}^*, & \text{if } i \neq j \end{cases} \tag{39}$$

By numerically evaluating this for randomly chosen pairs of points in datasets of benchmark systems, it was interesting to note that g^i of these points are very close to each other. Specifically, cosine similarity $c^i_{A,B}$ for gradients of randomly selected data point pairs (i.e., g^i_A gradient of S_i with respect to V for point A and g^i_B gradient for point B) can be measured by:

$$c^i_{A,B} = \cos(\theta^i_{A,B}) = \frac{Re(g^i_A \cdot {g^i_B}^*)}{||g^i_A||||g^i_B||}$$

The average similarity of randomly selected data pairs in the whole dataset $\mathcal{D}$ can be evaluated by:

$$c_{\mathcal{D}} = \mathbb{E}_{A,B \in \mathcal{D}} \mathbb{E}_{i \in \mathcal{N}} c^i_{A,B} \tag{40}$$

When $c_{\mathcal{D}}$ is nearly 1, it represents a nearly linear relationship between S_i and V [94]. This analytically leads to the linearity between S_i and $|V_i|$, as well as ϕ_i when phase angles are small, which is the common case in transmission networks [95]. This relationship extends to the near convexity of **[C6]** as well. We show this empirically for the 5 benchmark systems in different grid scales, including the IEEE 14-bus, IEEE 118-bus, PGLIB 118-bus, PEGASE 1354-bus, and PGLIB 2000-bus systems. The average similarity $c_{\mathcal{D}}$ computed for **[C3]** and **[C6]** for these systems is higher than 0.96. Additionally, the histogram of this similarity of 1.18 million points on the IEEE 118-bus system is plotted in Figure 20, showing a high score of this measurement close to 1.0.

Thus, the nearly linear and convex nature of the relationships between the inputs and outputs that correspond to constraints **[C3]** and **[C6]** as evident in common datasets [96],

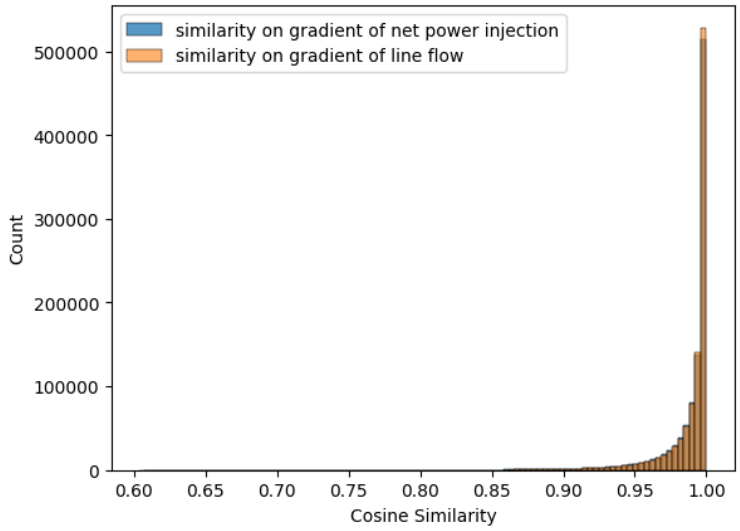

*Figure 20: Histogram of $c_{A,B}$ in IEEE 118-bus system for **[C3]** and **[C6]**.*

has motivated the convexification of $\mathcal{P}_{ACOPF}$ defined in Definition 3 and the use of ICNN. ICNN is more general than linear models and thus will be able to account for the non-linearities that are still present, and its convex properties allow for guarantees in solutions computed for the transformed problem. It is shown that ICNN is able to generalize well and approximate the feasible boundary defined by **[C3]** and **[C6]** with high accuracy in Section 6.4.

Defining a Convex Optimization Problem In order to eliminate the non-convexities presented in the constraints of the original OPF problem, a new function $H(x, y; \theta)$ is trained to classify if the input pairs of variables (x, y) are feasible or not. θ represents the parameters of the trained model. x encapsulates all variables in $\mathcal{P}_{ACOPF}$ (i.e., P_{G_g}, Q_{G_g}, $|V_i|$, and ϕ_i for all $g \in \mathcal{G}$ and $i \in \mathcal{N}$), and y is the composite of power demand, i.e., P_{D_i} and Q_{D_i} for all $i \in \mathcal{N}$. The output of $H(x, y; \theta)$ is a non-positive value if $\{x, y\}$ is feasible (i.e., satisfies constraints **[C1]** to **[C6]**) and a positive value otherwise. $H(x, y; \theta)$ will replace **[C1]** to **[C6]** in $\mathcal{P}_{ACOPF}$. The new problem will be referred to as $\mathcal{P}_{Conv}$ and is formulated as follows:

$$\mathcal{P}_{Conv} : \min_{P_{G_g}} \sum_{g \in \mathcal{G}} C_g(P_{G_g})$$

$$\text{s.t. } H(x, y; \theta) \leq 0$$

$$\forall g \in \mathcal{G} : P_{G_g}^{min} \leq P_{G_g} \leq P_{G_g}^{max}$$

$$Q_{G_g}^{min} \leq Q_{G_g} \leq Q_{G_g}^{max}$$

$$\text{s.t.} \forall i \in \mathcal{N} : V_i^{min} \leq |V_i| \leq V_i^{max}$$

$$\phi_i^{min} \leq \phi_i \leq \phi_i^{max}$$

With the convexity of $\mathcal{P}_{Conv}$ [94], many desirable attributes can be guaranteed in the computation of the solution of the ACOPF problem, including the convergence of iterative searches and global optima. Furthermore, this decouples the constraints from the objective and allows for the use of datasets that contain only operational points that are not necessarily optimal. Unlike convex relaxations, which are typically the supersets of the feasible set of the original problem and result in violations of grid limits, ICNN is trained to closely fit the feasible boundary via adversarial training. This approach is also different from approximation-based techniques that neglect important system attributes.

6.3.2 Data Augmentation and Training of ICNN

The model $H(x; \theta)$ is utilized to classify specific instances of grid variables x as being feasible or not. For this, it is necessary to utilize datasets composed of both feasible and infeasible data points to train the ICNN. However, in datasets that are available in practice, infeasible points are rare occurrences as these can lead to grid violations and cascading failures. System operators maintain a high degree of reliability in the operation of the grid [97]. To overcome

the lack of infeasible points, the dataset is augmented with infeasible points computed via adversarial perturbations of feasible points. Unlike image applications, small perturbations of data points in the power domain can render these infeasible. As such, after the augmentation process, labels are assigned whereby the original data points are assigned the value 0 and the augmented data points are each assigned the value 1.

The data augmentation technique proposed for the power grid application is different from those commonly utilized in image processing applications. With this approach, feasible points in the original dataset are randomly selected and perturbations are applied until they cross the decision boundaries formed by **[C1]-[C6]** in Definition 3. Four specific types of perturbations are utilized to generate the augmented data points. The first is based on additive noise, where for a feasible point x :$\{P_{G_g}$, Q_{G_g}, $|V_i|$, and ϕ_i, $\forall g \in \mathcal{G}$, $\forall i \in \mathcal{N}\}$ from the dataset, additive Gaussian noise with mean 0 and standard deviation of σ is applied to q_i. From empirical studies, σ is selected to be 0.02. The second technique is based on the swapping of variable set x between randomly selected power demand y. The third method utilizes uniform sampling of the input component of the feasible data point inside its valid range. The fourth method aims to generate data points violating **[C6]**. The gradient ascent algorithm is utilized to iteratively perturb x until **[C6]** is not satisfied.

With the augmented dataset, the ICNN is trained to be a classifier and identify whether the inputs are feasible (i.e., original data points) or not (i.e., augmented data points). During the training, the activation function at the output layer is selected to be the sigmoid function, which is non-convex. The sigmoid function allows for the use of the cross-entropy loss function specified in Equation 41 during the training process:

$$\begin{aligned} J(\theta) = &- \frac{1}{M} \sum_{m=1}^{M} U_m log[\sigma(H(x_m, y_m; \theta))] \\ &+ (1 - U_m) log[1 - \sigma(H(x_m, y_m; \theta))] \end{aligned} \tag{41}$$

where M denotes the total number of training points, in which half are randomly selected from the dataset and another half are from the data augmentation. σ is the sigmoid activation function and U_m returns the label assigned to point $\{x_m, y_m\}$ (i.e. 0 if $\{x_m, y_m\}$ is from the original dataset and 1 if $\{x_m, y_m\}$ is augmented). Once the training is complete, the sigmoid function is removed. When the output of $H(x_m, y_m; \theta)$ is non-positive, $\{x_m, y_m\}$ is feasible and otherwise it is inferred to be infeasible.

6.3.3 Iterative Search Algorithm

Once the ICNN is trained, $\mathcal{P}_{Conv}$ can be solved instead of $\mathcal{P}_{ACOPF}$. Since $\mathcal{P}_{Conv}$ is a convex formulation, the convergence to global optimum is guaranteed. First, a log-barrier-based objective function $\mathcal{L}$ is defined in Equation 42 to convert $\mathcal{P}_{Conv}$ into an unconstrained problem.

$$\begin{aligned}\mathcal{L}_{Conv}(x) = \sum_{g \in \mathcal{G}} C_g(P_{G_g}) - \tau[\log(-H(x, \gamma; \theta)) \\ + \sum_{i=1}^{K} (\log(x_i^{max} - x_i) + \log(x_i - x_i^{min}))]\end{aligned} \tag{42}$$

The first term in $\mathcal{L}_{Conv}(x)$ is the original objective function in $\mathcal{P}_{ACOPF}$, the second term is a log-barrier function that returns infinity when $\{x, y\}$ is infeasible. The log-barrier function acts as an indicator function and allows for a differentiable boundary. τ is a positive parameter that represents the weight of the log-barrier term. The log-barrier term takes higher precedence for lower values of τ and then diminishes over time. This allows for an iterative search to assign greater penalty for infeasibilities and then account for optimality over time. A log-barrier-based objective function is used instead of a Lagrangian dual as the linear penalty terms do not prevent the solution from being infeasible. The solution to $\mathcal{P}_{Conv}$ is computed in an iterative manner by minimizing $\mathcal{L}_{Conv}(x)$. The iterative updates

are defined as follows:

$$\begin{aligned} x_{t+1} =& x_t - \alpha(\beta \frac{\partial \mathcal{L}_{Conv}(x_t)}{\partial x_t} \\ &+ (1-\beta)\frac{\partial \mathcal{L}_{Conv}(x_{t-1})}{\partial x_{t-1}}) \end{aligned} \tag{43}$$

where t denotes the t-th iteration, α is a parameter representing the step-size in the update process, β is the momentum factor for accelerating the convergence process and $\frac{\partial \mathcal{L}_{Conv}(x)}{\partial x}$ is the gradient of $\mathcal{L}_{Conv}(x)$ which is computed by taking the gradient of the quadratic cost function and the ICNN model.

For this iterative algorithm, the initial point selected is critical, as otherwise the search will lead to out-of-distribution solutions. A warm-start approach is utilized where another neural network is trained to predict the feasible x given the input parameters y to the problem. This warm start point is further refined so that there is a margin of at least ν within the feasible boundary using the following iterative refinement:

$$x_0 = x_0 - \frac{\mathrm{d}H(x_0, y; \theta)}{\mathrm{d}x_0} \tag{44}$$

This refinement is repeated until $H(x_0, y; \theta) < \nu$. Starting from this initial point, x is iteratively refined until the stopping criteria of $H(x, y; \theta)$ return a positive value (i.e., x is infeasible) or $\frac{\partial \mathcal{L}_{Conv}}{\partial x} < \epsilon$, which indicates that the solution is near optimality and ϵ is a pre-defined threshold. Since Equation 43 represents updates to a convex loss function $\mathcal{L}_{Conv}$, the stopping criteria are guaranteed to be reached.

6.3.4 Feasibility Correction

The output x_{opt} of the searching process will be globally optimal for $\mathcal{P}_{conv}$. However, as $H(.)$ is a data-driven decision boundary, it is associated with a small margin of error. As a result, the solution x_{opt} from the iterative search, while globally optimal for the convex problem $\mathcal{P}_{conv}$, may slightly violate constraints **[C3]** and/or **[C6]** in the original formulation $\mathcal{P}_{ACOPF}$ in Definition 3. In experiments, the average aggregate imbalance in **[C3]** is approximately 0.01 p.u. for the benchmark systems, indicating that x_{opt} is very close to the feasible boundary. To address this, two stages of iterative refinements—local and global updates—are introduced to ensure the final solution meets **[C3]** and **[C6]** simultaneously under typical power grid conditions. These refinements will be detailed in the following sections.

Local Updates In this section, the proposed local updates are introduced. The residue corresponding to the power balance equation for bus i is defined as:

$$\mathcal{R}_i = P_{G_i} - P_{D_i} + j(Q_{G_i} - Q_{D_i}) - V_i \sum_{j \in \mathcal{N}} V_j^* Y_{ij}^* \tag{45}$$

Here, $\mathcal{R}_i$ is zero when there are no violations in the power balance equations, and non-zero otherwise. The local updates aim to reduce the residue to zero. For a generator bus i, the local power generation is updated once to absorb the local imbalance as follows:

$$P_{G_i}^{e+1} + jQ_{G_i}^{e+1} \leftarrow P_{D_i} + jQ_{D_i} + V_i^e \sum_{j \in \mathcal{N}} V_j^{e*} Y_{ij}^* \tag{46}$$

where e and $e+1$ denote the epoch (iteration) count for the global update. V_i^e and V_j^e are computed in the previous epoch e. At $e = 0$, these variables are initialized to x_{opt}. If

there is no slack in the generation constraints, bus i is treated as a load bus. For a load bus i, the updates are as follows:

$$V_{i,k+1}^{e+1} = \frac{1}{Y_{ii}} \left[\frac{-P_{D_i} + jQ_{D_i}}{V_{i,k}^{e+1*}} - \sum_{j=1, j\neq i}^{N} V_j^e Y_{ij} \right] \tag{47}$$

The subscripts k and $k+1$ refer to the local iteration count. Voltages from neighboring buses are computed in the previous epoch e (i.e., global update). These local updates k at the current epoch $e+1$ are derived by setting Equation 45 to zero, corresponding to the original constraint **[C3]**. The updates in Equation 47 applied to V_i follow from fixed-point iteration. These iterations are repeated until the residual calculated at iteration k falls below a predefined threshold ρ (i.e., $|\mathcal{R}_{i,k}^{e+1}| < \rho$) where ρ is chosen to be a very small value (e.g., 10^{-6}). Since the voltage values of the neighboring buses are from the previous epoch, the local updates can be performed in parallel by each bus. The convergence properties of these local updates are discussed in Theorem 6.1.

Theorem 6.1. *The local updates to bus voltages defined in Equation 47 are guaranteed to converge when the following holds:*

$$\left| \frac{-P_{D_i} + jQ_{D_i}}{Y_{ii}} \right| < |V_i^{min}|^2 \tag{48}$$

Proof. This proof follows from Banach's fixed-point theorem, which indicates that the iterative refinements must represent a contraction mapping for the sequence to converge to an equilibrium point that satisfies $\mathcal{R}_i = 0$.

When $\mathcal{R}_i = 0$, we have:

$$-P_{D_i} - jQ_{D_i} = V_i \sum_{j \in \mathcal{N}} V_j^* Y_{ij}^*$$

To derive the fixed point iterations, the above relation is rearranged in terms of V_i as follows:

$$-P_{D_i} - jQ_{D_i} = V_i V_i^* Y_{ii}^* + V_i \sum_{j \in \mathcal{N}, j \neq i} V_j^* Y_{ij}^*$$

$$V_i^* = \frac{1}{Y_{ii}^*}\Big(\frac{-P_{D_i} - jQ_{D_i}}{V_i} - \sum_{j \in \mathcal{N}, i \neq j} V_j^* Y_{ij}^*\Big)$$

$$V_i = \frac{1}{Y_{ii}}\Big(\frac{-P_{D_i} + jQ_{D_i}}{V_i^*} - \sum_{j \in \mathcal{N}, i \neq j} V_j Y_{ij}\Big)$$

The voltage variable on the left hand side of the above relation is treated as the updated variable and the voltage variables on the right hand side are obtained from computations in the previous iteration to obtain Equation 47.

In the proof of Theorem 1, first, the updates in Equation 47 are expressed as $f(V_i^k) = V_i^{k+1}$ as the update at iteration $k+1$ is a function of the voltage values computed in the previous iteration. This mapping is contractive if:

$$\Big|\frac{f(V_{i,k}) - f(V_{i,k-1})}{V_{i,k} - V_{i,k-1}}\Big| < 1$$

This is indeed the case for the fixed point iterations defined in Equation 47. Substituting the definition of $f(V_{i,k})$ into the left-hand side of the above inequality and rearranging the terms:

$$\frac{f(V_{i,k}) - f(V_{i,k-1})}{V_{i,k} - V_{i,k-1}} = \Big(\frac{-P_{D_i} + jQ_{D_i}}{Y_{ii}}\Big)\Big(\frac{\frac{1}{V_{i,k}} - \frac{1}{V_{i,k-1}}}{V_{i,k} - V_{i,k-1}}\Big)$$

$$= \Big(\frac{-P_{D_i} + jQ_{D_i}}{Y_{ii}}\Big)\Big(\frac{V_{i,k-1} - V_{i,k}}{V_{i,k} V_{i,k-1}(V_{i,k} - V_{i,k-1})}\Big)$$

Applying the magnitude operator to the above relation, the following derivations are obtained:

$$
\begin{aligned}
|\frac{f(V_{i,k}) - f(V_{i,k-1})}{V_{i,k} - V_{i,k-1}}| &\leq \Big|\frac{-P_{D_i} + jQ_{D_i}}{Y_{ii}}\Big|\Big|\frac{V_{i,k-1} - V_{i,k}}{V_{i,k}V_{i,k-1}(V_{i,k} - V_{i,k-1})}\Big| \\
&= \Big|\frac{-P_{D_i} + jQ_{D_i}}{Y_{ii}}\Big|\Big|\frac{1}{V_{i,k}V_{i,k-1}}\Big| \leq \Big|\frac{-P_{D_i} + jQ_{D_i}}{Y_{ii}}\Big|\frac{1}{|V_i^{min}|^2}
\end{aligned}
$$

The last inequality in the above stems the observation that all voltage values computed will heed constraints **[C4]** and **[C5]**. In typical power systems, the lowest possible magnitude of variables $V_{i,k}V_{i,k-1}$ can take will be 0.95^2 given that the upper and lower bounds are $V_i^{min} = 0.95$ and $V_i^{max} = 1.05$ [98] . For the fixed point iterations to converge, the above must be bounded by 1 resulting in:

$$
\Big|\frac{-P_{D_i} + jQ_{D_i}}{Y_{ii}}\Big| < |V_i^{min}|^2 = 0.9025
$$

□

Global Convergence Once local updates are complete, the values computed in the current epoch are exchanged with all neighbouring nodes and the updates defined in Equations 46 and 47 are repeated for the next epoch. This process is repeated until the magnitude of the overall residuals computed at each epoch is lower than a threshold (i.e. $\sum_{i\in\mathcal{N}} |R_i^e| \leq \rho$) or the number of epochs have reached a maximum of E. Global convergence across the epochs is also guaranteed and this is formalized in Theorem 6.2.

Theorem 6.2. *The local feasibility corrections defined in Equations 46 and 47 are guaranteed to globally converge across the epochs when the following condition is satisfied for all* $e \in \mathbb{E}$*:*

$$\sum_{i\in\mathcal{N}}(\sum_{j\in\mathcal{N},j\neq i}|\frac{c_i^e V_j^e Y_{ij}^*}{Y_{ii}^* V_i^e}|+\sum_{j\in\mathcal{L},j\neq i}|\frac{c_i^e c_j^e \mathcal{R}_j^e Y_{ij}^*}{Y_{ii}Y_{jj}^* V_i^{e*} V_j^e}|)|\mathcal{R}_i^e|<\sum_{i\in\mathcal{N}}|\mathcal{R}_i^e| \tag{49}$$

where c_i^e is the contraction factor of local updates at bus i in epoch e (defined in the proof) and $\mathbb{E}$ represents the set of all epochs. R_i^e is the residual at bus i calculated using voltage values updated at epoch e.

Proof. The condition listed in Theorem 2 is derived here. First, the total residual $\mathcal{R}^{e+1}$ across the grid at epoch $e+1$ is defined:

$$\begin{aligned}
&|\mathcal{R}^{e+1}|=\sum_{i\in\mathcal{N}}|\mathcal{R}_i^{e+1}|\\
=&\sum_{i\in\mathcal{G}}|P_{G_i}^{e+1}-P_{D_i}+j(Q_{G_i}^{e+1}-Q_{D_i})-V_i^e\sum_{j\in\mathcal{N}}(V_j^e+\Delta_{Vj}^{e+1})^*Y_{ij}^*|\\
+&\sum_{i\in\mathcal{L}}|-P_{D_i}-jQ_{D_i}-(V_i^e+\Delta_{Vi}^{e+1})\sum_{j\in\mathcal{N}}(V_j^e+\Delta_{Vj}^{e+1})^*Y_{ij}^*|\\
=&\sum_{i\in\mathcal{G}}|V_i^e\sum_{j\in\mathcal{N},j\neq i}\Delta_{Vj}^{e+1*}Y_{ij}^*|+\sum_{i\in\mathcal{L}}|(V_i^e+\Delta_{Vi}^{e+1})\sum_{j\in\mathcal{N},j\neq i}\Delta_{Vj}^{e+1*}Y_{ij}{}^*|
\end{aligned}$$

where Δ_{Vi}^{e+1} is the change in the bus voltage at bus i after applying local updates. The second and third lines in the above follow from splitting the residuals into generator buses and load buses and applying Equation 45 to these terms. When bus i is the generator bus, the power generation will be revised to meet power balance according to Equation 46 and if it is a load bus, the voltage is updated according to Equation 47. The fourth line results from updates in Equation 46 that lead to:

$$P_{G_i}^{e+1} - P_{D_i} + j(Q_{G_i}^{e+1} - Q_{D_i}) - V_i^e \sum_{j \in \mathcal{N}} V_j^{e*} Y_{ij}^* = 0$$

and applying Equation 47 to the load bus:

$$-P_{D_i} - jQ_{D_i} - (V_i^e + \Delta_{Vi}^{e+1}) \sum_{j \in \mathcal{N}} V_j^{e*} Y_{ij}^* = 0$$

Next, the change in voltage magnitude $\Delta_{V_i}^{e+1}$ at each load bus due to local updates from fixed point iteration is estimated. Firstly, it starts with the difference resulting from the voltage computed in epoch e and the first local update iteration based on Equation 47:

$$\begin{aligned} V_{i,1}^{e+1} - V_i^e &= \frac{1}{Y_{ii}}[\frac{-P_{D_i} + jQ_{D_i}}{V_i^{e*}} - \sum_{j=1, j \neq i}^{N} V_j^e Y_{ij}] - V_i^e \\ &= \frac{1}{Y_{ii} V_i^{e*}}[-P_{D_i} + jQ_{D_i} - V_i^{e*} \sum_{j=1}^{N} V_j^e Y_{ij}] \\ &= \frac{1}{Y_{ii} V_i^{e*}}[-P_{D_i} - jQ_{D_i} - V_i^e \sum_{j=1}^{N} V_j^{e*} Y_{ij}^*]^* = \frac{R_i^{e*}}{Y_{ii} V_i^{e*}} \end{aligned}$$

The right-hand side of the first line above follows from applying one update based on Equation 47 and replacing that update with $V_{i,1}^{e+1}$. In the second line, V_i^e is absorbed into the fixed point update. In the third line, the relation is re-arranged so that it can be expressed in terms of the residual from voltages computed at epoch e. The subsequent local updates based on fixed point iterations will result in the total change of voltage in that epoch amounting to be a factor c_i^e of the change resulting from the first iteration:

$$\Delta_{V_i}^{e+1} = \frac{c_i^e \mathcal{R}_i^{e*}}{Y_{ii} V_i^{e*}}$$

This factor c_i^e takes a value of approximately 1 on average (i.e. number of local updates are close to one). Then, this expression for the change is substituted into the expression of $|\mathcal{R}^{e+1}|$ as follows:

$$
\begin{aligned}
|\mathcal{R}^{e+1}| &= \sum_{i\in\mathcal{G}} |-V_i^e \sum_{j\in\mathcal{N},j\neq i} \frac{c_j^e \mathcal{R}_j^e Y_{ij}^*}{Y_{jj}^* V_j^e}| + \sum_{i\in\mathcal{L}} |-(V_i^e + \frac{c_i^e \mathcal{R}_i^{e*}}{Y_{ii} V_i^{e*}}) \\
&\sum_{j\in\mathcal{N},j\neq i} \frac{c_j^e \mathcal{R}_j^e Y_{ij}^*}{Y_{jj}^* V_j^e}| \\
&\leq \sum_{i\in\mathcal{G}} |V_i^e \sum_{j\in\mathcal{N},j\neq i} \frac{c_j^e \mathcal{R}_j^e Y_{ij}^*}{Y_{jj}^* V_j^e}| + \sum_{i\in\mathcal{L}} |V_i^e \sum_{j\in\mathcal{N},j\neq i} \frac{c_j^e \mathcal{R}_j^e Y_{ij}^*}{Y_{jj}^* V_j^e}| \\
&+ |\sum_{i\in\mathcal{L}} \frac{c_i^e \mathcal{R}_i^{e*}}{Y_{ii} V_i^{e*}} \sum_{j\in\mathcal{N},j\neq i} \frac{c_j^e \mathcal{R}_j^e Y_{ij}^*}{Y_{jj}^* V_j^e}| \\
&= \sum_{i\in\mathcal{N}} |V_i^e \sum_{j\in\mathcal{L},j\neq i} \frac{c_j^e \mathcal{R}_j^e Y_{ij}^*}{Y_{jj}^* V_j^e}| + |\sum_{i\in\mathcal{L}} \frac{c_i^e \mathcal{R}_i^{e*}}{Y_{ii} V_i^{e*}} \sum_{j\in\mathcal{L},j\neq i} \frac{c_j^e \mathcal{R}_j^e Y_{ij}^*}{Y_{jj}^* V_j^e}| \\
&\leq \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{L},j\neq i} |\frac{c_j^e V_i^e \mathcal{R}_j^e Y_{ij}^*}{Y_{jj}^* V_j^e}| + \sum_{i\in\mathcal{L}} \sum_{j\in\mathcal{L},j\neq i} |\frac{c_i^e c_j^e \mathcal{R}_i^{e*} \mathcal{R}_j^e Y_{ij}^*}{Y_{ii} Y_{jj}^* V_i^{e*} V_j^e}| \\
&= \sum_{i\in\mathcal{L}} \sum_{j\in\mathcal{N},j\neq i} |\frac{c_i^e V_j^e \mathcal{R}_i^e Y_{ij}^*}{Y_{ii}^* V_i^e}| + \sum_{i\in\mathcal{L}} \sum_{j\in\mathcal{L},j\neq i} |\frac{c_i^e c_j^e \mathcal{R}_i^{e*} \mathcal{R}_j^e Y_{ij}^*}{Y_{ii} Y_{jj}^* V_i^{e*} V_j^e}| \\
&= \sum_{i\in\mathcal{L}} (\sum_{j\in\mathcal{N},j\neq i} |\frac{c_i^e V_j^e Y_{ij}^*}{Y_{ii}^* V_i^e}| + \sum_{j\in\mathcal{L},j\neq i} |\frac{c_i^e c_j^e \mathcal{R}_j^e Y_{ij}^*}{Y_{ii} Y_{jj}^* V_i^{e*} V_j^e}|) |\mathcal{R}_i^{e*}|
\end{aligned}
$$

The original relation is rearranged so that the final expression in the above is a weighted summation of the magnitude of residual at every load bus. The condition in Theorem 6.2 requires the residuals in epoch $e+1$ to be lesser than the residuals in epoch e. When this holds for every epoch, this contractive behaviour will lead to converge to an equilibrium where all residuals are 0. □

Inequality Correction As mentioned earlier, the log-barrier terms allow for an implicit slack in the inequality constraints for the solutions computed using Equation 43. For in-

stance, in the IEEE 118-bus system, the slack is 0.72 p.u. for **[C6]** and 6.3% in the voltage limits. Thus, when feasibility correction is performed to rectify a very small imbalance, the resulting solutions do not violate any of the inequality constraints as observed in the experimental studies.

However, when inequality constraints are violated, for example, using the corrective method on the solution from SDP relaxations, the effects of the feasibility correction algorithm on the inequality constraints are analyzed. The violation of inequality constraints can be defined as a loss function in the form of:

$$\begin{aligned}\mathcal{L}_{Ineq} = &\sum_{i=1}^{K} \max(x_i - x_i^{max}, 0) + \max(x_i^{min} - x_i, 0) \\ &+ \sum_{i,j \in \mathcal{N}, j \neq i} \max(|V_i(V_i^* - V_j^*)Y_{ij}^*| - S_{i,j}^{max}, 0)\end{aligned} \tag{50}$$

where x_i represents all K decision variables, and the first summation term calculates the violation of box constraints defined in [C1], [C2], [C4] and [C5]. The second summation term quantifies the non-adherence of the electrical line limits in [C6]. Then, a gradient-based update will conduct iteratively on each variable x_i via the following rule, where ζ is the step size:

$$x_i \leftarrow x_i - \zeta \frac{\partial \mathcal{L}_{Ineq}}{\partial x} \tag{51}$$

Then, the algorithm needs to recheck the adherence of equality constraints and update control variables via Equations 46 and 47. These two methods alternate and will eventually achieve both equality and inequality constraints, as the search is conducted within the boundaries learned from the training data distribution, which obey all constraints.

As such, the proposed algorithm to compute a solution for $\mathcal{P}_{ACOPF}$ is summarized in Algorithm 3. The algorithm will exit and reject the query when it fails to find a valid starting point x_0 in line 6, and when no convergence is achieved in line 17.

Algorithm 3 Computation of solution for ACOPF.

1: **function** MAIN($F(\cdot)$, $H(\cdot;\theta)$, P_D, Q_D, ν, ϵ, ρ, M, T, E, K, Z)
2: Define $\gamma : \{P_D, Q_D\}$
3: Define $x : \{P_G, Q_G, |V_G|, |V_{\mathcal{N}\setminus\mathcal{G}}|, \phi_{\mathcal{N}}\}$
4: $x_0 \leftarrow F(\gamma)$
5: $m \leftarrow 0$
6: **while** $H(x_0, \gamma; \theta) > \nu$ **and** $m < M$ **do**
7: $m \leftarrow m + 1$
8: Update x_0 via gradient descent
9: **end while**
10: $t \leftarrow 0$
11: **while** $t < T$, $H(x_0, \gamma; \theta) < 0$ **and** $\frac{\partial \mathcal{L}_{Conv}}{\partial x} > \epsilon$ **do**
12: $t \leftarrow t + 1$
13: Update x_t with Equation 43
14: **end while**
15: $x_{opt} \leftarrow x_t$
16: $z \leftarrow 0$
17: **while** Not Converged **and** $z < Z$ **do**
18: CORRECTION(x_{opt})
19: Update x_{opt} with Equation 51
20: **end while return** x_{opt}
21: **end function**
22: **function** CORRECTION(x)
23: $e \leftarrow 0$
24: **while** $\sum_{i\in\mathcal{N}} |R_i^e| > \rho$ **and** $e \leq E$ **do**
25: $e \leftarrow e + 1$
26: **for each** i $\in \mathcal{N}$ **do**
27: **if** i is a load bus **then**
28: **while** $\left|\mathcal{R}_{i,k}^e\right| > \rho$ **and** $k < K$ **do**
29: $k \leftarrow k + 1$
30: Update $V_{i,k}^e$ with Equation 47
31: **end while**
32: $k \leftarrow 0$
33: **else**
34: Update $P_{G,i}^e$ and $Q_{G,i}^e$ with Equation 46
35: **end if**
36: **end for**
37: **end while**
38: **end function**

Compare to GS algorithm, the proposed calibration algorithm updates in parallel, and the local convergence is guaranteed by Theorem 6.1. Moreover, GS algorithm fixes the active power generation and voltage magnitude, and only updates reactive power and phase angle on each generator bus. The proposed method, however, fully absorbs the imbalance by changing both active and reactive power generations. The global convergence of the algorithm is supported by Theorem 6.2.

6.4 Experiments

In this section, the efficacy of the proposed algorithm is evaluated via practical and comparative studies. First, the manner in which the datasets used to train the ICNN model are generated is outlined in Section 6.4.1. Thereafter, the parameter selection, including the architecture of the ICNN model and all parameters used in this algorithm, are detailed in Section 6.4.2. Section 6.4.3 illustrates the convergence properties of the iterative algorithms proposed for solving $\mathcal{P}_{Conv}$ and feasibility correction. Subsequently, a comparative study of existing work is presented in Section 6.4.4.

6.4.1 Datasets

In the experiments, five benchmark systems are associated, including IEEE 14-bus, IEEE 118-bus, PEGASE 1354-bus, PGLIB 118-bus, and PGLIB 2000-bus systems. The IEEE 14-bus system is composed of 14 buses, 5 generators, and 11 loads. The IEEE 118-bus and PGLIB 118-bus systems consist of 19 generators, 35 synchronous condensers, 177 lines, 9 transformers, and 91 loads [40, 99]. The power flow constraints on electrical lines for IEEE bus-118 have been adopted from reference [100]. The PEGASE 1354-bus system is composed of 1,354 buses, 260 generators, and 1,991 branches. The PGLIB bus-2000 systems contain 2,000 buses, 384 generators, and 3,639 lines.

All datasets used in the training and evaluation of the proposed algorithm have been generated using MatPower. Power demands are sampled within [80%,120%] of nominal values for four benchmark systems: IEEE 14-bus, IEEE 118-bus, PEGASE 1354-bus, and PGLIB 2000-bus systems. To simulate the drastic variation of power demand, the sample range for PGLIB bus-118 is as large as [50%,150%]. To achieve the best performance, the dataset should have sufficient variety and quantity in the sampling range, and the demand of testing cases should fall within the same range. This influences the model's performance by enabling not only a close approximation of the feasible boundary but also accurate prediction of the starting point in the convex optimization process. The total number of data points generated are 244,489, 226,986, 200,322, 188,332, and 156,012 for the IEEE 14-bus, IEEE 118-bus, PGLIB 118-bus, PEGASE 1354-bus, and PGLIB 2000-bus benchmark systems, respectively. Each dataset is divided into three parts where 10,000 points are randomly selected for testing, and the remaining 75%, 15% and 10% are used for training, validation and testing.

6.4.2 Selection of Parameters

Architecture Selection The selection of the number of layers, nodes per layer, optimizer, and learning parameters for a data-driven model is an empirical process that requires systematic search. These parameters dictate the performance of the underlying model that is trained and thus are a critical process. As the underlying grid structures are fundamentally different among the five benchmark systems, a separate model $H(x, \gamma; \theta)$ is trained for each case. Table 7 details various combinations of architectures and model parameters explored for the IEEE bus-14, IEEE bus-118, and PEGASE bus-1354 systems. Figures 21-23 illustrate the evolution of prediction accuracy of each H model on the validation dataset as the training progresses for the various settings listed in Table 7. The settings with † in Table 7

are selected for the final H models as these exhibit low losses, rapid convergence, and greater stability during training (i.e., less oscillations). The model selection process for the other two power grids is similar.

Table 7: Architecture of the Proposed ICNN Model

System	Setting	#Layers & Neurons	#Parameters	Batch	Optimizer	LR	Decay
IEEE14	1	66|128|64|32|4|1	50.8k	512	Adam	$1e^{-3}$	-
	2	66|128|64|32|4|1	50.8k	512	Adam	$1e^{-2}$	0.95/100
	3†	66|256|128|64|16|8|1	156.9k	512	Adam	$1e^{-3}$	$0.95/1,000$
	4	66|512|128|64|16|1	420.2k	512	Adam	$1e^{-2}$	$0.95/1,000$
	5	66|256|128|64|16|1	156.2k	512	SGD	$1e^{-3}$	$0.95/1,000$
IEEE118	1	580|256|128|64|32|4|1	536.6k	512	Adam	$1e^{-3}$	-
	2	580|256|128|64|32|4|1	536.6k	512	Adam	$1e^{-3}$	$0.95/1,000$
	3	580|256|128|64|32|4|1	536.6k	512	SGD	$1e^{-3}$	$0.6/1,000$
	4	580|256|128|64|32|4|1	536.6k	128	Adam	$1e^{-3}$	$0.8/1,000$
	5	580|256|128|64|32|16|1	546.5k	512	Adam	$1e^{-2}$	$0.9/1,000$
	6†	580|512|256|128|64|16|1	1.30m	512	Adam	$1e^{-3}$	$0.95/1,000$
PEGASE1354	1	5936|256|128|64|16|8|1	4.43m	32	Adam	$1e^{-3}$	$0.9/1,000$
	2	5936|1024|512|128|64|32|16|1	18.27m	8	Adam	$1e^{-3}$	$0.95/1,000$
	3†	5936|512|128|64|16|1	7.65m	16	SGD	$1e^{-3}$	$0.9/1,000$
	4	5936|128|64|32|16|8|1	3.26m	16	SGD	$1e^{-3}$	$0.95/1,000$

Parameters in Algorithm 3 Parameters and pre-defined thresholds selected for Algorithm 3 are presented in Table 8. Figure 24 presents the effect of various values of the parameter τ encountered in Equation 42 on the outputs of a log barrier term. It is clear that larger values of τ impose a penalty in a damped manner (i.e., less steep) when the inputs are closer to 0. In the iterative updates listed in Equation 43, the values of τ are adjusted in a scheduled manner across three stages as listed in Table 8. The threshold ϵ for checking the convergence of convex optimization is set to $1e^{-6}$ for all five systems. The parameter K for local convergence, ρ and E for global convergence in the feasibility correction methods are set to 10, $1e^{-6}$, and 100, respectively. The remaining parameters are empirically selected to facilitate rapid convergence to equilibrium for each of the benchmark systems considered.

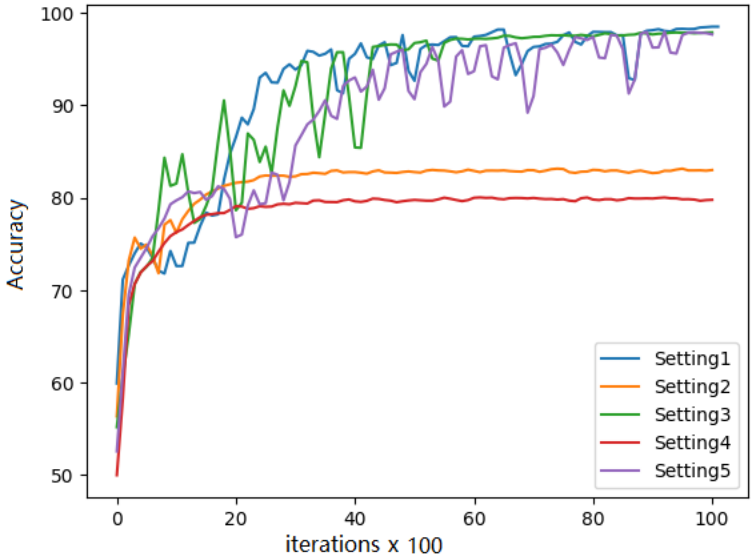

Figure 21: Performance of ICNN Models on Validation Set for IEEE 14-bus system.

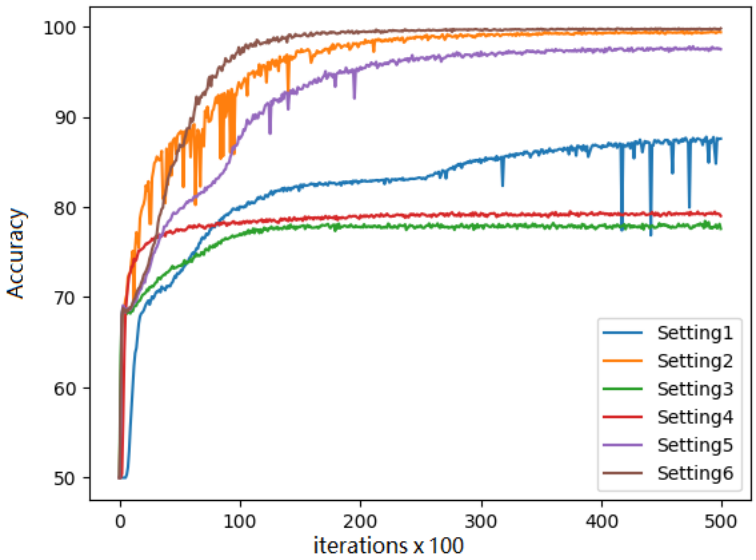

Figure 22: Performance of ICNN Models on Validation Set for IEEE 118-bus system.

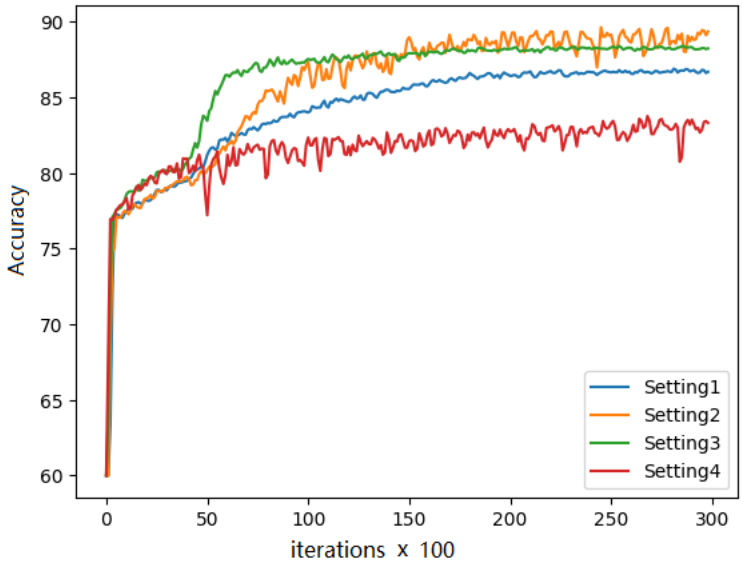

Figure 23: Performance of ICNN Models on Validation Set for PEGASE 1354-bus system.

Table 8: Parameters in Alg. 3.

Parameter	τ	ν	α	β
IEEE14	$[1e^{-2}, 1e^{-3}, 1e^{-4}]$	-0.35	$1e^{-6}$	0.6
IEEE118	$[1e^{-3}, 5e^{-4}, 1e^{-4}]$	-0.5	$1e^{-5}$	0.5
PGLIB118	$[1e^{-2}, 1e^{-3}, 1e^{-4}]$	-0.5	$1e^{-5}$	0.6
PEGASE1354	$[1e^{-2}, 1e^{-3}, 1e^{-4}]$	-0.5	$1e^{-6}$	0.6
PGLIB2000	$[1e^{-2}, 1e^{-3}, 1e^{-4}]$	-0.35	$1e^{-5}$	0.6

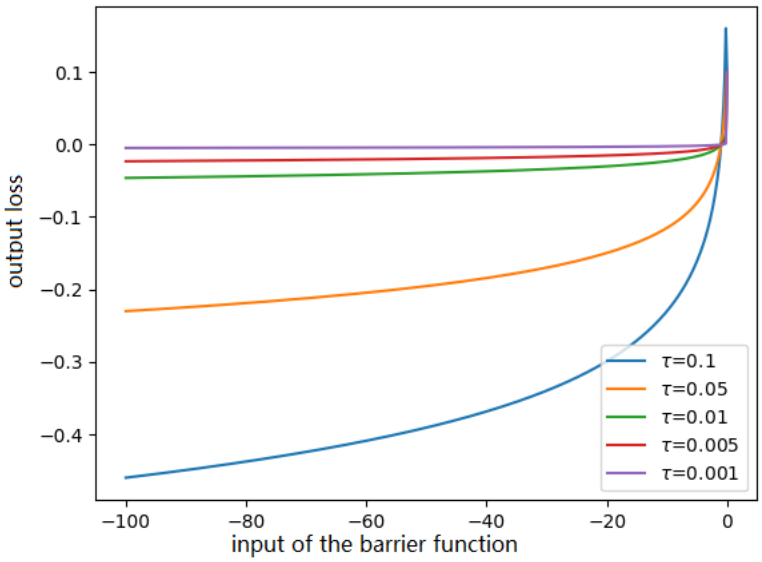

Figure 24: Logarithm Barrier Functions with Different Values of τ

6.4.3 Convergence Studies

Here, we study the convergence properties of the three iterative computations introduced in Algorithm 3: 1) Iterative computation of the solution to $\mathcal{P}_{Conv}$; 2) Local feasibility updates guaranteed to converge based on the condition in Theorem 1; and 3) Global convergence supported by the condition listed in Theorem 2.

Convergence to the Convex Optimization Problem As $\mathcal{P}_{Conv}$ is a convex optimization problem, the iterative updates defined in Equation 43 are guaranteed to converge to global optima as long as the interior of the feasible space is non-empty. The iterative updates entail the computation of the gradient of $H(.)$ which is a convex function but can be complex depending on the number of layers and nodes used to construct the model. We use the built-in auto-differentiation module available in TensorFlow and the Keras deep learning library to compute the gradient of $H(.)$ [102]. This function is optimized to rapidly compute gradients of complex machine learning models. For instance, the time taken to compute the gradient of $H(.)$ in the 1354-bus system is 0.016s on average. In Figure 25, an example of

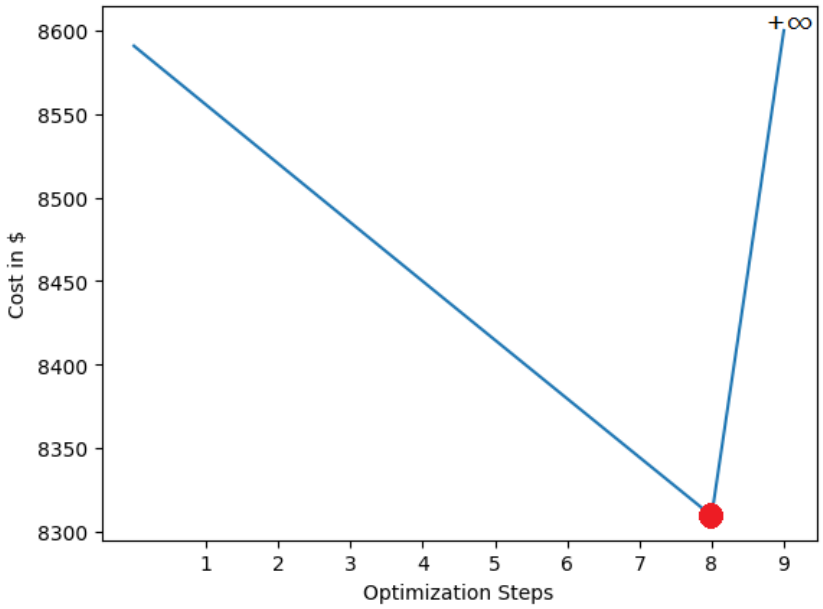

Figure 25: Evolution of Search on $\mathcal{P}_{Conv}$.

how $\mathcal{L}_{conv}$ evolves over the iterative updates for a randomly selected instance γ is presented. It is clear that starting from the selected initial point, the cost evolves to drop until the optimum is found. The objective function in $\mathcal{P}_{Conv}$ is monotonic regarding the active power generation $P_{G_g}, g \in \mathcal{G}$. Consequently, the optimization process drives the variables to their boundaries in search of the feasible solution with the lowest cost. Thus, the update will eventually become infeasible at which point the value of $\mathcal{L}_{conv}$ shoots to infinity, and the step before this infeasible point is the optimum. x_{opt} is selected to be the point computed right before the infeasible update. The convergence is fast as depicted in Figure 25. The average numbers of iterations required for convergence are: 6.4, 8.7, 7.7, 10.2, and 11.4 for the 5 systems respectively.

Convergence of Theorem 1 The condition (i.e., Equation 48) listed in Theorem 1 must be satisfied for the local fixed-point updates listed in Equation 47 to converge. For the IEEE bus14 and IEEE bus118, the largest values taken by the left-hand side of the condition are

0.093 and 0.077, respectively. These fall well below the threshold listed on the right-hand side of Equation 48, which is 0.95^2. As such, this is verified for every point in the datasets where local iterations converge 100% of the time. The average number of iterations required for convergence is 1.49.

Convergence of Theorem 2 In order to ensure global convergence, the condition (i.e., Equation 49) listed in Theorem 2 must be satisfied. We show that this is the case for all power grids considered in this paper by first observing the characteristics of the coefficient of $\mathcal{R}_i^e$ in Equation 49, which will be denoted as a_i:

$$a_i = \begin{cases} \sum_{j \in \mathcal{N}, j \neq i} |\frac{c_i^e V_j^e Y_{ij}^*}{Y_{ii}^* V_i^e}| + \sum_{j \in \mathcal{L}, j \neq i} |\frac{c_i^e c_j^e \mathcal{R}_j^e Y_{ij}^*}{Y_{ii} Y_{jj}^* V_i^{e*} V_j^e}| & i \in \mathcal{L} \\ 0 & i \in \mathcal{G} \end{cases}$$

In IEEE 14-bus, IEEE 118-bus, PEGASE 1354-bus and PGLIB 2000-bus systems; 35%, 46%, 19% and 19% of buses are generator buses respectively. PGLIB 118-bus has 19% of buses that can generate both active and reactive power, 46% of buses can only generate reactive power. These result in the corresponding proportion of a_i terms taking 0 values. When evaluating a_i in experimental studies, the minimum and maximum nonzero values taken by a_i is tabulated in Table 9 for the five systems along with the percentage of instances where a_i takes values lesser than 1. It is clear that for all testing systems, the majority of a_i computed are less than 1 and when this occurs, the corresponding residuals will be lower in the next epoch. However, when this is not the case, since most coefficients are lesser than 1, close to 1, or are 0, the overall residual across the grid reduces across each epoch. This is verified experimentally using all five datasets and convergence was observed for every point evaluated (i.e., approximately 200,000 points in each dataset). Figure 26 illustrates the evolution of the residual for a randomly selected instance in the 118-bus system. It is clear that convergence occurs very rapidly and that the overall residual across the grid approaches

0 within 15 epochs and exhibits no oscillatory behaviour.

Table 9: Convergence Analysis of Theorem 2

Grid	GEN(%)	$\lvert a_i \rvert$ RANGE	$\lvert a_i \rvert < 1$	CON.(✓/×)
IEEE 14	35%	[0.335,1.009]	92%	✓
IEEE 118	46%	[0.541,1.012]	85%	✓
PGLIB 118	16%,46%	[0.0007,1.004]	99%	✓
PEGASE 1354	19%	[0.003,1.008]	99%	✓
PGLIB 2000	19%	[0.0007,1.004]	99%	✓

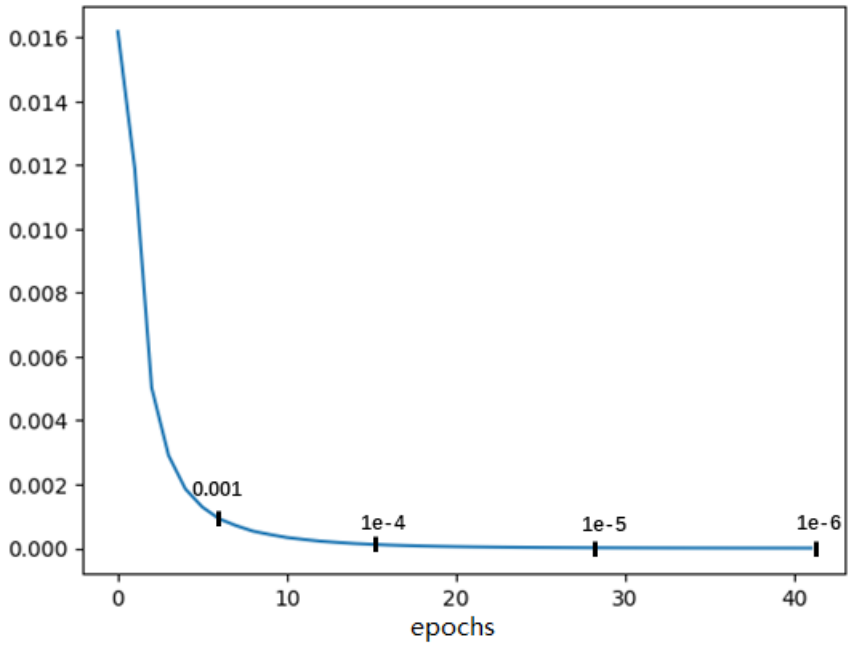

Figure 26: Global Convergence Curve of Theorem 2

After the convergence of the proposed corrective steps, the cost gap between the corrected solution of P_{Conv} and P_{ACOPF} is lower than 0.5% across all tested systems. To further reduce this gap, it is possible to restrict or eliminate the capacity of high-cost generators to absorb imbalance by changing their update type to load buses or only adjusting reactive power. We leave the further experimental and theoretical convergence study with limited capacity of generators as a direction for future work.

6.4.4 Comparative Studies

In this section, a comparative study is conducted to compare this proposal with recent proposals in the literature that aim to solve the ACOPF problem via data-driven approaches and other common convex relaxation techniques. Solutions obtained from the Matlab Interior Point Solver (MIPS) serve as the benchmark.

First, the performance of the following data-driven approaches, along with our proposal, are compared: reinforcement learning (RL) [103], generative adversarial networks (GANs) [68], graph neural networks (GNN) [21, 64], and the DC3 algorithm [65, 66]. Next, we compare the performance of our algorithm with other convex relaxation-based techniques. Common approaches in the literature include second-order cone programming (SOCP) [35, 36] and semi-definite programming (SDP) relaxations [16]. According to reference [34], SDP relaxations are tighter than SOCP relaxations for electric grids with a mesh topology (as is the case for general power systems). The results tabulated in Table 10 include: percentage deviation of the computed solutions from that of the MIPS solver (i.e., column % MIPS) along with the 95% confidence interval, whether the equality constraint and inequality constraints are met by the solutions (i.e., columns **Eq.** and **InEq.**), and finally, the average time taken to compute the solution. Test datasets curated for each benchmark system are utilized for the results listed in Table 10.

The MIPS solver requires less than 0.5 seconds for the IEEE 14-bus, IEEE 118-bus, and PGLIB 118-bus systems, where all solutions are verified to be feasible. For the Pegase 1354-bus and PGLIB 2000-bus systems, however, the computational times are higher, and the convergence rate drops to 59.2% and 87.3%. The RL-based technique claims optimality but does not provide guarantees on feasibility. The GAN-based solver [68] converges rapidly; however, with no guarantees on optimality and feasibility. The DC3 [65] performs feasibility corrections by utilizing the Newton-Raphson solver. The computational time for their al-

Table 10: Comparative Study with Existing Work.

Alg	% MIPS	Eq.	InEq.	Time (s)
IEEE14				
MIPS [40]	-	✓	✓	0.25
PROPOSED	1.53± 0.27	✓	✓	0.08
SDP [16]	$1.7e^{-3}$	×	×	0.36
RL [103]	0.6	×	×	-
IEEE118				
MIPS	-	✓	✓	0.30
PROPOSED	1.97± 0.18	✓	✓	0.12
GAN [68]	3.5	×	✓	0.5
SDP	$2.6e^{-2}$	×	×	7.68
DC3 [66]	0.3	✓	×	-
GNN [64]	8	×	×	-
GNN2 [21]	5	×	-	-
PGLIB118				
MIPS	-	✓	✓	0.36
PROPOSED	1.58± 0.25	✓	✓	0.16
SDP	$1.51e^{-3}$	×	×	14.82
SOCP [104]	4.69	×	×	19.93
PEGASE1354				
MIPS	-	✓	✓	2.58
PROPOSED	1.66± 0.20	✓	✓	0.32
GAN	2.6	×	✓	1.03
SDP	×	×	×	152.7
PGLIB2000				
MIPS	-	✓	✓	3.42
PROPOSED	2.37± 0.34	✓	✓	0.55
SDP	×	×	×	367.9
SOCP	×	×	×	120.3

gorithm is comparable to ours. However, this method provides no guarantees on inequality constraints. The unsupervised GNN [64] method is associated with significantly higher optimality gaps. Another GNN-based proposal in [21] has slightly lower optimality gaps, which are still significantly higher than other methods. The solutions computed by SDP are very close to those computed by the MIPS solver, with no guarantees on feasibility due to the relaxations (as no guarantees exist for mesh topology in transmission systems). SOCP [104] generates solutions for PGLIB 118-bus with a 4.69% optimality gap and no guarantees on feasibility as well and failed to converge for large systems such as PGLIB 2000-bus. Furthermore, the computational costs for both SDP and SOCP are high (i.e., up to 2 minutes for the 1354-bus system and 2000-bus systems). Our proposal, on the other hand, is verified to always be feasible and rapidly (i.e., within 0.35 seconds for the 1354-bus system) produces optimal solutions that are close to the benchmark solver (within 1 to 2%). It is important to note that we did not require datasets with optimal solutions to achieve these results.

6.5 Summary

In this chapter, a data-driven approach is proposed to solve the ACOPF problem in a tractable manner, and a physics-based feasibility correction algorithm is also included that is proven to converge under non-restrictive conditions. This proposal allows for the use of existing operational data points available in abundance to grid operators, which are feasible but not necessarily optimal. The efficacy of the proposed framework is demonstrated for practical IEEE 14-bus, IEEE 118-bus, PGLIB 118-bus, PEGASE 1354-bus, and PGLIB 2000-bus systems. The proposed algorithm can generate solutions that maintain pre-set feasibility criteria while producing near-optimal solutions. Comparative studies also demonstrate the performance of our algorithm in comparison to the state-of-the-art.

7 Conclusion and Future Work

The modern electrical grid is facing a transition from traditional physical system to a smarter grid with advanced technologies and new resources. At the core of power grid operations, OPF problem is essential for determining the most cost-effective dispatches in the power grid while adhering all system constraints. This thesis proposes novel machine learning-based approaches that aim to overcome the limitations of traditional OPF solvers, particularly with respect to computational efficiency, feasibility, the handling of non-convexity, and decoupling of traditional iterative solvers. This section summarizes the key advancements made in this thesis and outlines potential future research directions for further extending these innovations.

7.1 Summary of the Thesis

In this thesis, an overview of the power grid and its operational challenges is illustrated first, followed by an introduction to the basics of power grids and power flow studies. This includes two models for representing power balance, the formulation, and the roles of the optimal power flow (OPF) problem. Due to the non-convex nature of the OPF problem, traditional solvers and relaxation methods cannot solve it effectively within an acceptable time frame. A large body of data-driven methods have been proposed in the literature to shift the computational burden from online to offline. In this thesis, three novel machine learning-based algorithms are proposed, and their contributions are listed below:

Leveraging domain knowledge The proposed algorithms incorporate topology information and physics equations from power flow studies as additional knowledge, beyond just data. These are integrated into physics-informed loss functions and calibration techniques.

This novel integration not only enables machine learning models to converge faster and more accurately but also improves the feasibility of the solutions generated by the proposed OPF solvers.

Relaxing the need for optimal dataset In proposed GAN-based and ICNN-based OPF solvers in Chapter 5 and 6, the reliance on generating optimal OPF solutions through traditional solvers is reduced. Instead, historical operational datasets from ISO data centers are utilized to learn feasible data distributions and search for the optimum within the learned boundaries.

Novel Machine Learning Methods for optimization Representation learning and convex neural networks are specifically tailored and applied to the domain of power systems in this thesis. These methods provide insights into using machine learning to explore power flow data and perform optimizations based on the knowledge acquired through the learning process.

Feasibility corrections In all three proposals, algorithms are derived to post-process the output of the OPF solvers to ensure feasibility. These corrections are necessary due to the highly nonlinear constraints in the OPF formulation. Theoretical proof supports the convergence properties of the proposed methods.

7.2 Future Work

While the machine learning-based solutions proposed in this thesis address several limitations of traditional Optimal Power Flow (OPF) solvers, there are still various areas that warrant further exploration. The following sections outline key future research directions, focusing

on enhancing the adaptability, scalability, and practicality of OPF solutions in real-world power systems. These include addressing topology changes, improving the scalability of data-driven methods, and incorporating discrete variables into the optimization process.

7.2.1 OPF on Topology Change

Current data-driven solvers primarily operate on fixed topologies, where connectivity and line parameters remain unchanged across scenarios. However, these can vary in practice, and data-driven methods need to adapt to such conditions while still guaranteeing optimality and feasibility. Topology changes can result from unit commitment decisions, scheduled changes, and contingencies or emergencies.

Topology Change due to Unit Commitments Unit commitment is a key process in the day-ahead market, determining the dispatchability of each generator for the following days. When a generator is dispatchable, the bus it connects to is considered a generator bus in the OPF formulation; otherwise, it is treated as a load bus. Consequently, these decisions can significantly alter the network topology and, therefore, the OPF results. However, similar unit commitment schedules may occur due to factors such as time of day, day of the week, season, or weather conditions. As a result, there could be a finite set of common topologies corresponding to different unit commitment patterns. Separate data-driven models can be trained for these patterns to solve the OPF problem effectively.

Scheduled Topology Changes Topologies in transmission networks do not change very often as they are designed to remain stable over long periods of time. Nevertheless, there are scheduled alternations such as upgrade of transmission lines, installation of new lines, etc. In these cases, small batch of optimal power flow data are possible to be generated to assist the knowledge transfering in the context of transfer learning or model adaption. It will be

promising to discover the impact of OPF solutions from different type of changes.

Topology Change in Emergency Emergent topology changes such as transmission line failure, generator outages can be caused by extreme weathers, mechanical/fuel issues, malicious behaviour, etc. To handle these random event for OPF, the research needs to be extended to Security-Constrained Optimal Power Flow (SCOPF). SCOPF evaluates the system's response to k potential contingencies out of N components in the grid (e.g., "N-1 SCOPF" or "N-k SCOPF" , meaning loss of one or more system components in the decision process of SCOPF). It is interesting to explore the equilibrium between the robustness and optimality of SCOPF, because the output of SCOPF is commonly more conservative than OPF due to the consideration of contingencies.

7.2.2 Scalability of Data-driven OPF

Data-driven OPF accelerates traditional OPF solvers by shifting the computational burden from online optimization to offline learning. However, as the power grid grows, applying machine learning models becomes increasingly challenging. This is primarily due to the numerous variables (e.g., voltage, power generation, load) and constraints (e.g., power balance, generation limits, transmission limits). Additionally, the amount of data required to adequately cover the solution space grows exponentially in high-dimensional spaces, leading to computation and generalization challenges. Two potential directions to address scalability issues include:

Distributed Optimization Decomposing complex optimization problems into sub-problems that can be solved locally is a promising extension of OPF research. It would be a fundamental contribution to investigate different region-splitting mechanisms and their impact on distributed optimization results. More importantly, due to the non-convex nature of

ACOPF, seeking theoretical convergence would be significant.

Dimensionality Reduction Due to the large number of constraints and their complexity, variables in OPF are highly interdependent. This supports the manifold hypothesis, which states that high-dimensional real-world data often lie on much lower-dimensional manifolds. Exploring low-dimensional representations of power flow data, whose dimensionality does not increase exponentially with grid size, is a promising avenue for research.

7.2.3 OPF with Discrete Variables

This thesis assumes all variables are continuous, changing smoothly within their value range. However, in practice, some variables are discrete. For example, tap ratios can be controllable variables, with the best tap setting chosen to minimize the objective function. Another extension involving discrete variables is Optimal Transmission Switching, where transmission lines can be either connected or disconnected based on the optimization problem. The existence of discrete variables transforms the OPF problem into a Mixed Integer Nonlinear Programming (MINLP) problem with higher non-convexity and discrete feasible regions. Data-driven methods offer promising potential to reduce the search space by classifying the states of discrete variables, predicting warm-start points in the potential search region, or reducing dimensionality to avoid convergence issues associated with traditional MINLP solvers.

7.3 Conclusion

This chapter has provided a comprehensive overview of the key contributions made in this thesis. These advancements have demonstrated improved computational efficiency, better handling of non-convexity, and enhanced feasibility of solutions. However, the evolving

nature of the power grid continues to present new challenges. As discussed in the future work section, there are several promising research directions that could further improve the practicality and scalability of data-driven OPF solutions. In conclusion, while this thesis contributes significant progress in the field of OPF, future research is essential to further address the challenges in modern power systems. The proposed future directions offer a path toward more robust and scalable solutions, enabling the grid to operate more efficiently in the face of increasing uncertainty and variability.

Appendices

A Modeling of Distributed Energy Resources

The dataset used in this thesis incorporates power generation from DERs, specifically wind power and photovoltaic power. In this section, the modeling of both types of DERs is elaborated.

A.1 Modeling of Wind Power

The amount of electricity generated from wind farm at any given moment depends on several factors: the wind speed v, average air density ρ, and the swept area of wind turbines A in the wind farm. The active power generated by wind can be calculated using the following equation [105]:

$$P_w = \frac{A\rho v^3}{2}$$

The air density ρ varies in the range of 1.15 to 1.4 $\mathrm{kg/m^3}$ under different temperatures [106], but in this thesis, it is considered constant at 1.225 $\mathrm{kg/m^3}$ (the density at 15°C).

The swept area of a wind turbine can be calculated as follows:

$$A = \pi R^2$$

where R is the radius of the wind turbine, ranging from 40 to 70 meters [107].

The wind speed changes over time, but it typically follows a Weibull distribution (or

Rayleigh distribution, a special case when $k = 2$), defined as:

$$f_w(\mathbf{v}) = \left(\frac{k}{c}\right)\left(\frac{\mathbf{v}}{c}\right)^{k-1} \exp\left[-\left(\frac{\mathbf{v}}{c}\right)^{k}\right]$$

where c is the annual mean wind speed at the location of the wind farm in meters per second (e.g., 4.16 m/s in Toronto [108]). The shape parameter k is selected to be 2 in this thesis because the resulting distribution covers a wide range of values, such as the case when wind speed reaches 16 m/s in extreme weather conditions. The distribution of the modeled wind speed is depicted as follows:

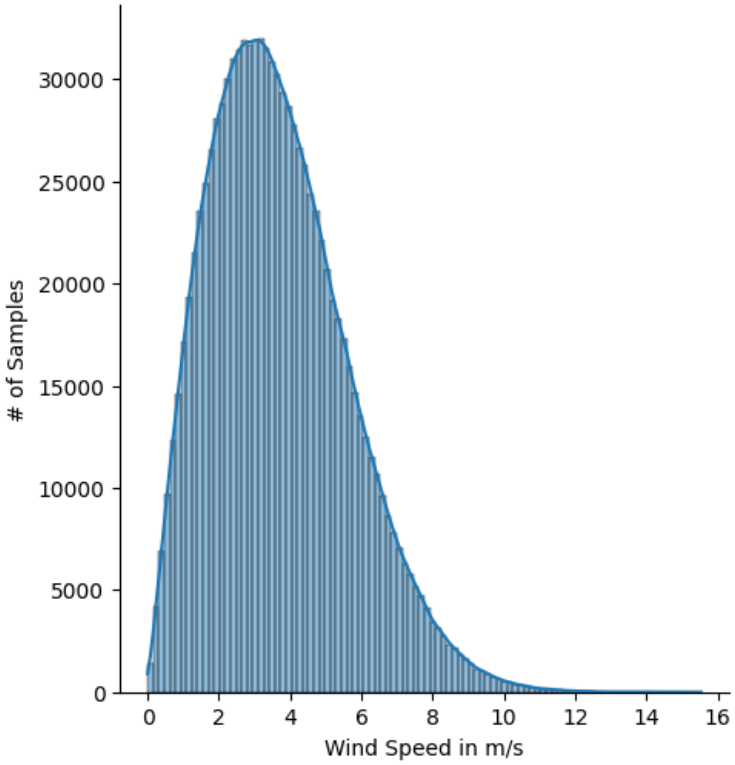

Figure 27: Distribution of the Modeled Wind Speed

In the simulation, due to the grid size and penetration rate of wind power, it is assumed that each wind farm has only one turbine with a swept area of 1800 m^2.

A.2 Modeling of Solar Power

The electricity generated by photovoltaic energy depends on the solar irradiance E, which is the amount of solar power per unit area received by the panel, measured in watts per square meter (W/m^2), and the area of the solar panel in the solar farm. The active power generated in a solar farm can be calculated by:

$$P_s = A \times E \times \eta$$

where η is the energy conversion efficiency, ranging from 17% to 20% in practice [109], and is set to be 20% consistently in this thesis.

In the literature, solar irradiance is commonly modeled using a Weibull distribution [110–112] with parameters average irradiance c and shape factor k.

$$f_s(\mathbf{E}) = \left(\frac{k}{c}\right)\left(\frac{\mathbf{E}}{c}\right)^{k-1} \exp\left[-\left(\frac{\mathbf{E}}{c}\right)^{k}\right]$$

The average irradiance in Toronto is 166.67 W/m^2 [113]. The dimensions of the most common commercial solar panels are 6.5 feet by 3 feet [114], which is 1.81 m^2, and a solar farm may possess thousands of panels. These parameters are scaled in the experiment to match the percentage of solar generation.

References

[1] Pirathiyini, Srikantha. "Introduction to Power Systems", Course Material of EECS3622, EECS, York University, Canada, 2021.

[2] Cain, Mary B., Richard P. O'neill, and Anya Castillo. "History of optimal power flow and formulations." Federal Energy Regulatory Commission 1 (2012): 1-36.

[3] "Weather-related disasters increase over past 50 years, causing more damage but fewer deaths" https://public.wmo.int/en/media/press-release/weather-related-disasters-increase-over-past-50-years-causing-more-damage-fewer

[4] "Climate Change: Global Temperature" https://www.climate.gov/news-features/understanding-climate/climate-change-global-temperature.

[5] https://www.un.org/en/climatechange/net-zero-coalition

[6] https://www.epa.gov/ghgemissions/sources-greenhouse-gas-emissions

[7] Sissine, Fred. "Energy Independence and Security Act of 2007: a summary of major provisions." Library of Congress Washington DC Congressional Research Service, 2007.

[8] Abdallah, Lamiaa, and Tarek El-Shennawy. "Reducing carbon dioxide emissions from electricity sector using smart electric grid applications." Journal of Engineering 2013 (2013).

[9] https://www.eia.gov/

[10] https://www.electricity.ca/

[11] https://www.ferc.gov/electric-power-markets

[12] https://www.iso-ne.com/about/what-we-do/in-depth/industry-standards-structure-and-relationships

[13] https://www.caiso.com/

[14] Lavaei, Javad, and Steven H. Low. "Zero duality gap in optimal power flow problem." IEEE Transactions on Power systems 27, no. 1 (2011): 92-107.

[15] S.J.Wright"Primal-dual interior-point methods". Society for Industrial and Applied Mathematics, 1997.

[16] SH.Low. "Convex relaxation of optimal power flow—Part I: Formulations and equivalence." IEEE Transactions on Control of Network Systems 1.1: 15-27, 2014.

[17] Li, Fangxing, and Rui Bo. "Small test systems for power system economic studies." In IEEE PES general meeting, pp. 1-4. IEEE, 2010.

[18] Grainger, John J. Power system analysis. McGraw-Hill, 1999.

[19] Kelley, Carl T. Solving nonlinear equations with Newton's method. Society for Industrial and Applied Mathematics, 2003.

[20] J.Carpentier. "Contribution a l'etude du dispatching economique." Bulletin de la Societe Francaise des Electriciens 3.1: 431-447, 1962.

[21] Liu, Shaohui, Chengyang Wu, and Hao Zhu. "Topology-aware graph neural networks for learning feasible and adaptive ac-opf solutions." IEEE Transactions on Power Systems (2022).

[22] Gayme, Dennice, and Ufuk Topcu. "Optimal power flow with large-scale storage integration." IEEE Transactions on Power Systems 28, no. 2 (2012): 709-717.

[23] Zhao, Tianyu, et al. "DeepOPF+: A deep neural network approach for DC optimal power flow for ensuring feasibility." 2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm). IEEE, 2020.

[24] Zhang, Ling, Yize Chen, and Baosen Zhang. "A convex neural network solver for dcopf with generalization guarantees." IEEE Transactions on Control of Network Systems (2021).

[25] Venzke, Andreas, et al. "Learning optimal power flow: Worst-case guarantees for neural networks." 2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm). IEEE, 2020.

[26] Deka, Deepjyoti, and Sidhant Misra. "Learning for DC-OPF: Classifying active sets using neural nets." 2019 IEEE Milan PowerTech. IEEE, 2019.

[27] Eldridge, Brent, Richard O'Neill, and Anya Castillo. "An improved method for the DCOPF with losses." IEEE Transactions on Power Systems 33.4 (2017): 3779-3788.

[28] Pan, Xiang, Tianyu Zhao, and Minghua Chen. "Deepopf: Deep neural network for dc optimal power flow." 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm). IEEE, 2019.

[29] Baker, Kyri. "Solutions of dc opf are never ac feasible." Proceedings of the Twelfth ACM International Conference on Future Energy Systems. 2021.

[30] "Recent ISO software enhancements and future software and modeling plans," Last accessed Feb. 28, 2022, [Online] Available: https://cms.ferc.gov/sites/default/files/2020-05/rto-iso-soft-2011.pdf.

[31] Winner, Calla, Jasmine Garland, Constance Crozier, and Kyri Baker. "Carbon Emissions Resulting from Different Power Flow Models for Dispatch." In 2023 IEEE Power & Energy Society General Meeting (PESGM), pp. 1-5. IEEE, 2023.

[32] Bedoya, Juan Carlos, et al. "A qcqp and sdp formulation of the optimal power flow including renewable energy resources." 2019 International Symposium on Systems Engineering (ISSE). IEEE, 2019.

[33] SH.Low. "Convex relaxation of optimal power flow—Part I: Formulations and equivalence." IEEE Transactions on Control of Network Systems 1.1: 15-27, 2014.

[34] Madrigal, Marcelino, and A. D. Quintana. "Semidefinite programming relaxations for 0, 1-power dispatch problems." 1999 IEEE Power Engineering Society Summer Meeting. Conference Proceedings (Cat. No. 99CH36364). Vol. 2. IEEE, 1999.

[35] Soofi, Arash Farokhi, Saeed D. Manshadi, Guangyi Liu, and Renchang Dai. "A SOCP relaxation for cycle constraints in the optimal power flow problem." IEEE Transactions on Smart Grid 12, no. 2 (2020): 1663-1673.

[36] Tian, Zhuang, and Wenchuan Wu. "Recover feasible solutions for SOCP relaxation of optimal power flow problems in mesh networks." IET Generation, Transmission & Distribution 13, no. 7 (2019): 1078-1087.

[37] Yasasvi, P. Naga, Snehil Chandra, A. Mohapatra, and S. C. Srivastava. "An exact SOCP formulation for AC optimal power flow." In 2018 20th National Power Systems Conference (NPSC), pp. 1-6. IEEE, 2018.

[38] Chowdhury, Md Mahmud-Ul-Tarik, Biswajit Dipan Biswas, and Sukumar Kamalasadan. "Second-order cone programming (SOCP) model for three phase optimal power flow (OPF) in active distribution networks." IEEE Transactions on Smart Grid (2023).

[39] Low, Steven H. "Convex relaxation of optimal power flow—Part II: Exactness." IEEE Transactions on Control of Network Systems 1, no. 2 (2014): 177-189.

[40] Zimmerman, Ray Daniel, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education." IEEE Transactions on power systems 26, no. 1 (2010): 12-19.

[41] Torres, Geraldo Leite, and Victor Hugo Quintana. "An interior-point method for non-linear optimal power flow using voltage rectangular coordinates." IEEE transactions on Power Systems 13, no. 4 (1998): 1211-1218.

[42] Capitanescu, Florin, Mevludin Glavic, Damien Ernst, and Louis Wehenkel. "Interior-point based algorithms for the solution of optimal power flow problems." Electric Power systems research 77, no. 5-6 (2007): 508-517.

[43] Casacio, Luciana, Christiano Lyra, and Aurelio RL Oliveira. "Interior point methods for power flow optimization with security constraints." International Transactions in Operational Research 26, no. 1 (2019): 364-378.

[44] Momoh, James A., and J. Z. Zhu. "Improved interior point method for OPF problems." IEEE Transactions on Power Systems 14, no. 3 (1999): 1114-1120.

[45] Fioretto, Ferdinando, Terrence WK Mak, and Pascal Van Hentenryck. "Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 01. 2020.

[46] R. Canyasse, G. Dalal, and S. Mannor, "Supervised learning for optimal power flow as a real-time proxy," in 2017 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), pp. 1–5, IEEE, 2017.

[47] Yang, Yan, et al. "Fast calculation of probabilistic power flow: A model-based deep learning approach." IEEE Transactions on Smart Grid 11.3 (2019): 2235-2244.

[48] Lei, Xingyu, Zhifang Yang, Juan Yu, Junbo Zhao, Qian Gao, and Hongxin Yu. "Data-driven optimal power flow: A physics-informed machine learning approach." IEEE Transactions on Power Systems 36, no. 1 (2020): 346-354.

[49] Owerko, Damian, Fernando Gama, and Alejandro Ribeiro. "Optimal power flow using graph neural networks." In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5930-5934. IEEE, 2020.

[50] Baker, Kyri. "Emulating ac opf solvers with neural networks." IEEE Transactions on Power Systems 37, no. 6 (2022): 4950-4953.

[51] Guha, Neel, Zhecheng Wang, Matt Wytock, and Arun Majumdar. "Machine learning for AC optimal power flow.", 2019.

[52] Pan, Xiang, Minghua Chen, Tianyu Zhao, and Steven H. Low. "DeepOPF: A feasibility-optimized deep neural network approach for AC optimal power flow problems." IEEE Systems Journal 17, no. 1 (2022): 673-683.

[53] Jia, Yujing, and Xiaoqing Bai. "A CNN approach for optimal power flow problem for distribution network." 2021 Power System and Green Energy Conference (PSGEC). IEEE, 2021.

[54] Zamzam, Ahmed S., and Kyri Baker. "Learning optimal solutions for extremely fast AC optimal power flow." In 2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pp. 1-6. IEEE, 2020.

[55] Albadi, Mohammed, and K. Volkov. "Power flow analysis." Computational Models in Engineering (2020): 67-88.

[56] Baker, Kyri. "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.

[57] Falconer, T., & Mones, L. (2022). Leveraging power grid topology in machine learning assisted optimal power flow. IEEE Transactions on Power Systems.

[58] John, Elizabeth, and E. Alper Yıldırım. "Implementation of warm-start strategies in interior-point methods for linear programming in fixed dimension." Computational Optimization and Applications 41, no. 2 (2008): 151-183.

[59] Rahman, Jubeyer, Cong Feng, and Jie Zhang. "Machine learning-aided security constrained optimal power flow." 2020 IEEE Power & Energy Society General Meeting (PESGM). IEEE, 2020

[60] Jia, Yujing, Xiaoqing Bai, Liqin Zheng, Zonglong Weng, and Yunyi Li. "ConvOPF-DOP: A data-driven method for solving AC-OPF based on CNN considering different operation patterns." IEEE Transactions on Power Systems 38, no. 1 (2022): 853-860.

[61] Huang, Wanjun, Xiang Pan, Minghua Chen, and Steven H. Low. "Deepopf-v: Solving ac-opf problems efficiently." IEEE Transactions on Power Systems 37, no. 1 (2021): 800-803.

[62] Singh, Manish K., Vassilis Kekatos, and Georgios B. Giannakis. "Learning to solve the AC-OPF using sensitivity-informed deep neural networks." IEEE Transactions on Power Systems 37, no. 4 (2021): 2833-2846.

[63] Huang, Wanjun, Minghua Chen, and Steven H. Low. "Unsupervised Learning for Solving AC Optimal Power Flows: Design, Analysis, and Experiment." IEEE Transactions on Power Systems (2024).

[64] Owerko, Damian, Fernando Gama, and Alejandro Ribeiro. "Unsupervised optimal power flow using graph neural networks." In ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6885-6889. IEEE, 2024.

[65] Donti, Priya L., David Rolnick, and J. Zico Kolter. "Dc3: A learning method for optimization with hard constraints." arXiv preprint arXiv:2104.12225 (2021).

[66] K.Chen, B.Shourya and Z.Yu. "Unsupervised Deep Learning for AC Optimal Power Flow via Lagrangian Duality." GLOBECOM 2022-2022 IEEE Global Communications Conference. IEEE, 2022.

[67] Wang, Junfei, and Pirathayini Srikantha. "Data-driven AC Optimal Power Flow with Physics-informed Learning and Calibrations." The 2024 SmartGridComm conference, Under Review.

[68] Wang, Junfei, and Pirathayini Srikantha. "Fast Optimal Power Flow With Guarantees Via an Unsupervised Generative Model." IEEE Transactions on Power Systems, 2022.

[69] Wang, Junfei, and Pirathayini Srikantha. "Data-driven Convex Relaxations for Optimal Power Flow with Feasibility Guarantees." IEEE Transactions on Power Systems, Under Review.

[70] Sayed, Ahmed Rabee, Cheng Wang, Hussein I. Anis, and Tianshu Bi. "Feasibility constrained online calculation for real-time optimal power flow: A convex constrained deep reinforcement learning approach." IEEE Transactions on Power Systems 38, no. 6 (2022): 5215-5227.

[71] Jiang, Hui. Machine learning fundamentals: A concise introduction. Cambridge University Press, 2021.

[72] Bishop, Christopher M., and Nasser M. Nasrabadi. Pattern recognition and machine learning. Vol. 4, no. 4. New York: springer, 2006.

[73] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

[74] Kingma, Diederik P., and Max Welling. "An introduction to variational autoencoders." Foundations and Trends® in Machine Learning 12, no. 4 (2019): 307-392.

[75] Song, Jiaming, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models." arXiv preprint arXiv:2010.02502 (2020).

[76] Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).

[77] Weng, Lilian. "From gan to wgan." arXiv preprint arXiv:1904.08994 (2019).

[78] Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784 (2014).

[79] X.Ding, et al. "CcGAN: Continuous Conditional Generative Adversarial Networks for Image Generation." International Conference on Learning Representations, 2020.

[80] Y.Zheng, Y.Zhang and Z.Zheng. "Continuous Conditional Generative Adversarial Networks (cGAN) with Generator Regularization.", 2021

[81] Chen, Xi, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." Advances in neural information processing systems 29 (2016).

[82] Kramer, Mark A. "Nonlinear principal component analysis using autoassociative neural networks." AIChE journal 37, no. 2 (1991): 233-243.

[83] Chen, Pin-Yu, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models." In Proceedings of the 10th ACM workshop on artificial intelligence and security, pp. 15-26. 2017.

[84] Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C. Courville. "Improved training of wasserstein gans." Advances in neural information processing systems 30 (2017).

[85] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks." In International conference on machine learning, pp. 214-223. PMLR, 2017.

[86] https://bestpracticeenergy.com

[87] Grainger, John J. Power system analysis. McGraw-Hill, 1999.

[88] Ghofrani, Mahmoud, Mohammad Hassanzadeh, Mehdi Etezadi-Amoli, and M. Sami Fadali. "Smart meter based short-term load forecasting for residential customers." In 2011 North American Power Symposium, pp. 1-5. IEEE, 2011.

[89] Short-term load forecasting in smart grid: A combined CNN and K-means clustering approach."

[90] Xia, Min, Haidong Shao, Xiandong Ma, and Clarence W. de Silva. "A stacked GRU-RNN-based approach for predicting renewable energy and electricity load for smart grid operation." IEEE Transactions on Industrial Informatics 17, no. 10 (2021): 7050-7059.

[91] Amos, Brandon, Lei Xu, and J. Zico Kolter. "Input convex neural networks." In International Conference on Machine Learning, pp. 146-155. PMLR, 2017.

[92] Chen, Yize, Yuanyuan Shi, and Baosen Zhang. "Data-driven optimal voltage regulation using input convex neural networks." Electric Power Systems Research 189 (2020): 106741.

[93] Salton, Gerard, Anita Wong, and Chung-Shu Yang. "A vector space model for automatic indexing." Communications of the ACM 18, no. 11 (1975): 613-620.

[94] Boyd, Stephen P., and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.

[95] Kundur, Prabha. "Power system stability." Power system stability and control 10 (2007): 7-1.

[96] http://www.cse.yorku.ca/p̃srikan/datasets.html

[97] JD.Glover, MS.Sarma and T.Overbye. "Power system analysis and design", SI version. Cengage Learning, 2012.

[98] C.Balu and D.Maratukulam. "Power system voltage stability.", New York, NY, USA: McGraw-Hill, 1994.

[99] S.Babaeinejadsarookolaee, et al. "The power grid library for benchmarking ac optimal power flow algorithms." , 2019.

[100] https://www.iit.edu/ [Accessed:27-March-2023]

[101] Y.Bengio. "Practical recommendations for gradient-based training of deep architectures." In Neural networks: Tricks of the trade: Second edition (pp. 437-478). Berlin, Heidelberg: Springer, 2012.

[102] B.Van Merriënboer, et al. "Automatic differentiation in ML: Where we are and where we should be going." Advances in neural information processing systems 31, 2018.

[103] Y.Zhou, et al. "A data-driven method for fast ac optimal power flow solutions via deep reinforcement learning.", Journal of Modern Power Systems and Clean Energy, 8(6), pp.1128-1139, 2020.

[104] B.Kocuk, et al. "Strong SOCP relaxations for the optimal power flow problem". Operations Research, 64(6), pp.1177-1196, 2016.

[105] A.Parajuli. "A statistical analysis of wind speed and power density based on Weibull and Rayleigh models of Jumla, Nepal." Energy and Power Engineering, 8(7), pp.271-282.

[106] R.Floors and M.Nielsen. "Estimating air density using observations and re-analysis outputs for wind energy purposes.", Energies 12, no. 11 (2019): 2038.

[107] https://www.energy.gov/eere/articles/wind-turbines-bigger-better [Accessed: 1-Sep-2024]

[108] https://toronto.weatherstats.ca/metrics/wind_gust_speed.html[Accessed: 1-Sep-2024]

[109] https://css.umich.edu/publications/factsheets/energy/photovoltaic-energy-factsheet [Accessed: 1-Sep-2024]

[110] M.U.Afzaal,et al. "Probabilistic generation model of solar irradiance for grid connected photovoltaic systems using weibull distribution.", Sustainability, 12(6), p.2241, 2020.

[111] H.H.Surendra, D.Seshachalam and K.R.Sudhindra. "Reliability analysis of solar energy resources using Weibull distribution for a standalone system in Indian context." Sciences, 7(1), 2020.

[112] L.Garbai and Z.Kovacs. “The estimation and forecast of solar energy yield with Weibull distribution.”, International Review of Applied Sciences and Engineering, 2021.

[113] https://www.solarenergylocal.com/states/south-dakota/toronto/ [Accessed: 1-Sep-2024]

[114] https://www.energysage.com/solar/average-solar-panel-size-weight/ [Accessed: 1-Sep-2024]