Learning to Operate a Sustainable Power System

Yize Chen

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2021

Reading Committee:

Baosen Zhang, Chair

Daniel S. Kirschen

Radha Poovendran

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

Learning to Operate a Sustainable Power System

Yize Chen

Chair of the Supervisory Committee:
Baosen Zhang
Electrical and Computer Engineering

The electric power system is undergoing dramatic transformations due to the emergence of renewable resources and demand-side revolutions. However, in order to face the increasing level of system complexity and uncertainty, we need to come up with algorithms that are able to operate the power grid in a safe, reliable and sustainable manner. Such algorithms need to be compatible with infrastructure advancements as well as societal needs. We argue in this thesis that data-driven methods combined with physical knowledge and optimization theories provide a set of powerful tools that can achieve performance guarantees in a diverse set of energy system applications.

This dissertation covers the uncertainty modeling, optimal control, and decision-making for power systems with high penetration of renewables. Leveraging the availability of sensing, actuation and data platforms, we are able to construct specifically-design data-driven algorithms by incorporating domain knowledge such as time series characterization, function convexity, and optimization problem structures. In particular, we present three algorithmic contributions in this work: we present a generative model to forecast possible future scenarios of renewable generators; we illustrate an optimal control framework which is built upon on input convex neural network; we formulate a machine learning task for the optimal power flow problem, and we come up with a generalizable and robust learning-based optimization solver to enable real-time decision-making under the environment of stochastic electricity demand. The development of such algorithms is built upon the foundation of control, optimization and machine learning theories.

In addition to these algorithmic contributions, a central focus of this thesis is the application of such data-driven tools in many challenging energy system tasks, while extending the state of the art. The developed methods have enabled more robust electric grid planning and energy storage operations. We also apply the neural network controller to voltage regulation and building energy management tasks, where in both domains the underlying systems are hard to model, and our approaches demonstrate the possibility of achieving optimal control with only sensing data available.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I have been incredibly fortunate and privileged throughout my entire life to have been given many opportunities that have led me to pursue this thesis research. Thanks to the thousands of people in our planet throughout the past few centuries who have provided me with the foundation, environment, knowledge, support, health, love, and happiness to produce this work.

This thesis has been the product of a close and fruitful collaboration with my advisor, Baosen Zhang. I remembered when I asked for his advice on PhD studies in our first meeting, he repeated three times: "Work hard!", which has motivated me through research challenges. I am hugely indebted to Baosen for all the help, guidance and insight he has provided to me without any reservation. Baosen's mindset, patience and passion have profoundly shaped the way I approach academic problems and pursue research directions, and more broadly I am fortunate to have learned much more from him along the way. His philosophy of looking at the simplest settings with the most intuitions has been molding into my research. In addition, his enthusiasm and happiness to talk about any new ideas have constantly made it a joy to work with him.

I want to thank the other members of my committee for their help with this work. Daniel Kirschen, in addition to providing detailed comments on this work, has for the past five years been an incredible mentor. I have learned many perspectives of our electric grids from his classes, emails, group meetings and online lectures. Radha Poovendran has been a constant source of inspiration, and in all my talks with him always ignites excitement about how the work we do in power system and algorithms can really change the world. I would also like to thank Cynthia Chen for giving many valuable suggestions regarding my career, and Lillian Ratliff and Sreeram Kannan for being on my qualify exam committee. I also want to say my appreciation for the faculty members at UW Clean Energy Institute, in particular Daniel Schwartz, David Ginger, and Shaun Taylor. Their

generous support and broad envisions have supported my PhD study.

I was very fortunate to work with many amazing minds at UW. I would like to express my gratitude to Yuanyuan Shi, who navigated me through a variety of domains from automatic control, machine learning to power systems. Talking with Yushi Tan and Hao Wang about various grid resilience issues and algorithms, both realistic and wildly implausible versions, has been one of the most fun parts of the past five years. Working with Yishen Wang, Ling Zhang, Jesus Elmer Contreras Ocana, and Pan Li were other highlights of my PhD experience. Their excitement in talking about novel ideas and conducting fundamental research has always been an inspiration. And thanks to everybody else at UW who have made this journey rewarding. This includes Yao Long, Chase Dowling, Tanner Fiez, Daniel Calderone, Ryan Elliott, Daniel Olsen, Lane Smith, Daniel Tabas, Wenqi Cui, Rahul Mallik, Atinuke Abolaji Ademola-Idowu, Abeer Almainouni, Mareldi Ahumada, Nina Vincent, Jackie Baum, Trisha Ray, Zeyu Wang, Daniel Olsen, Jiayi Li, Xiangyu Gao, Yan Chen, Qiangqiang Guo, Yuxuan Cheng, Kun Su, Miguel Ortega-Vazquez, Qingchun Hou, Bolun Xu, Yi Wang, Chenghui Tang, Hao Li, Liyuan Zheng, Xun Sun, Yueyang Chen, Baicen Xiao, Hossein Hosseini, Dinuka Sahabandu, Zhipeng Liu, Sang Sagong, Xiyu Wang, Cunzhi Ren, Chen Zou, Peifeng Jing, Yandi Shen, Yan Liang, Hao Geng, Namit Chauhan, Yuxuan Cheng, Yaxuan Zhou, Minghui Lu, Shan Lin, Bosong Li, Nam Song, and all lab alunmi we have met and worked together. A special thanks to our EE Dawg chatroom members Yue Sun, Yihan Jiang, and Xuhang Ying, as it was a blast going through the PhD rollercoaster together, and I am proud to see the amazing work you are doing.

My PhD would have been severely lacking without my internships at Los Alamos National Laboratory in 2018 and Microsoft Research in 2020. I learned many lifetime advice, graph theory, and control theory from Deepjyoti Deka and Misha Chertkov. I learned cutting-edge research on graph neural networks and large-scale machine learning engineering from Weiwei Yang and Ben Cutler. I am also grateful for all of the conversations and collaborations with the other interns and researchers in the industry as well. Besides, I was lucky to collaborate with Hao He and Sarang

Amirtabar from Centrica, Alimuddin Mohammad from Boeing, Uzma Siddiqi from Seattle City Light, Molly Shor and Mike Rea from E8 where I got to know the multiple paths and exciting research projects in industry.

Finally, moving away from the academic sphere, I can't imagine getting through the past several years without the support of family and friends. To my mom, Jing Wang, my dad, Xuanshe Chen, I want to thank you for all your support and encouragement. And thanks to everybody else who have made graduate school incredibly enjoyable. These wonderful memories will stay with me for life. This includes Jingkai Chen, Dmitri Shostakovich, Yuanjun Deng, Luyao Wang, Chengyang Liu, Chuliang Song, Quanlai Li, Bowen Zhao, Ruomeng Xu, Wanjiang Zheng, Zhendong Cao, Ludwig van Beethoven, Sergei Rachmaninoff, Qiushui Zhang, Fengnan Yue, Nan Zhang, Chi Zhang, Yilian Wang, Dingkang Wang, Shuo Zhang, Qihua Zhang, Yuexi Wang, Rui Zhao, Ruiyi Li, Sinong Geng, Bowen Li, Shaohui Liu, Yuqi Zhou, Umar Hashmi, Siddharth Bhela, Ronak Mehta, Shengwei Zhang, Peng Zhang, Boning Huang, Cheng Pan, Yize Sun, Bingqing Peng, Shijia Wei, Congmei Jiang and all my lifetime friends. I would like to especially thank Jingkai and Yize for their many helps, both professional and personal, over the years, dating back to our undergrad and high school years. Thank you all from the bottom on my heart.

DEDICATION

To everybody loves me.♡

Chapter 1

INTRODUCTION

Power and energy research is integral to building a clean and sustainable future. To achieve this goal, we need to integrate much higher penetration of renewables, to support the electrification of transportation, and to adapt to rapid growth of new participants such as data centers and energy storage. Consequently, the power system is becoming more complex and uncertain. For example, to combat climate change, California and New Jersey will require all passenger vehicles sold by 2035 are zero emission. Much of this new fleet would be charged via the electric grid, bringing tremendous changes to the landscape of electricity demand, while the aging grid itself is experiencing unprecedented challenges. From the other side of the picture, we have witnessed rolling blackouts among many other reliability problems of our energy systems, which calls for more advanced tools for power system modeling, planning and operation. Faced with the fast-changing landscape of renewable energy and demand-side evolution, we need to come up with efficient and reliable solutions in order to meet our objectives regarding climate change.

To achieve the climate and clean energy goals set by the various initiatives, and to tackle such emerging challenges in the engineering domains of electric grids, this dissertation combines a diverse set of tools from control theory, machine learning and optimization. By focusing on the rich interplay between theoretical foundations and new measurement data, this thesis focus on developing: (i) A novel data-driven tool for modeling and quantifying uncertainties in renewables generation; (ii) An efficient machine learning solver for power optimization problems; and (iii) A generalizable deep-learning control framework with optimality guarantees. Broadly, these tailored algorithms are designed for the modeling and control of cyber-physical energy systems.

1.1 Summary of Research Contributions

Machine learning can help improve the operation and control of power systems in several ways. From the generation side, it is often a concern on how to better dispatch the stochastic renewables generation. To overcome the difficulties of accurately modeling such future uncertainties, we propose a data-driven (or model-free) approach to generate possible future realizations by adopting *generative* methods. In Chapter 3, we describe how to train a deep neural network in a generative approach, such that the trained neural network is learning the underlying distribution of renewables generation, and can generate future scenarios. The content of Chapter 3 appears in [2, 3, 4]. Such scenario generation approach is also adopted in several applications proposed by us [5, 6, 7].

Chapter 4 addresses the challenge of designing optimal controller faced with unknown system dynamics. It turns out the key is to carefully design machine learning architectures that leverage the physics of the systems. We leverage the data obtained from sensor measurements to construct an input convex neural network (ICNN) [8] for system dynamics modeling, where the weights between neurons are constrained to be positive and some direct "passthrough" layers are added for better representation power. Without loss of generality, we prove that ICNNs can represent all convex system dynamics and are exponentially more efficient in fitting nonconvex systems than piecewise affine functions. Using the proposed ICNN, we show that many real-world control problems in energy systems (e.g., building energy management, and distribution system voltage control) can be cast as convex optimization problems, leveraging the prior knowledge that the underlying physics is convex. The content of Chapter 4 on constructing neural-network-based optimal controller mainly appears in [9, 10]. In [11, 12] we take a specific look into the voltage regulation problem in distribution grids, and extend the control framework to the distributed setting.

Another fundamental problem in power systems is OPF, in which the optimal combination of generator dispatch is calculated based on variable demand profile. Due to the fact that OPF needs to be recaculated at every time instance (e.g., 15 minutes), and the underlying optimization problem is often dealing with a large number of nodal and network constraints, it is of research interests to develop fast and accurate OPF solver. We find it is possible to learn the mapping from variable load

inputs to active (binding) constraints, which is able to greatly improve the solution time of a family of OPF problem. Such algorithms also have favorable generalization and robustness guarantees, which hold the promises for using machine learning as an efficient and reliable optimization solver. In Chapter 5, we describe the proposed machine learning techniques for learning to solve OPF, and the major results are illustrated in [13, 14].

There are also non-thesis research which are not discussed in details in this dissertation. In [15], we take a quick response with regards to the COVID-19 pandemic, and come up with a solution for accurately forecasting the electricity demand. In [16], we make use of the generative model to tackle the link prediction challenge for a set of graph problems. We also look into the emerging blockchain technology along with its potential applications in social networks [17]. Another direction we explore is machine learning security [18, 19, 20], where adversarial examples may be crafted to cause severe physical impacts for real-world systems.

Chapter 2

PRELIMINARIES AND BACKGROUND

This chapter provides a broad overview of foundational ideas and background material related to this thesis. We will also include a discussion of the related literature. Some knowledge of the general power system models, control algorithms and machine learning tasks will also be covered. The content in this thesis builds on the following topics. We assume preliminary knowledge of these topics and give a limited set of key references here.

2.1 Learning to Model the Physical Systems

Decisions on how to best operate and control complex physical systems such as the power grid, commercial and industrial buildings, transportation networks and robotic systems are of critical societal importance. These systems are often challenging to control because they tend to have complicated and poorly understood dynamics, sometimes with legacy components are built over a long period of time [21]. Therefore detailed models for these systems may not be available or may be intractable to construct. For instance, since buildings account for 40% of the global energy consumption [22], many approaches have been proposed to operate buildings more efficiently by controlling their heating, ventilation, and air conditioning (HVAC) systems [23]. Most of these methods, however, suffer from two drawbacks. On one hand, a detailed physics model of a building can be used to accurately describe its behavior, but this model can take years to develop. On the other hand, simple control algorithms have been developed by using linear (RC circuit) models [24] to represent buildings, but the performance of these models may be poor since the building dynamics can be far from linear [25].

Recent years have seen the surge of data-driven approaches for system modeling. More recently, there has been a strong push to further incorporate structured prediction methods like energy-based

architectures [26], while some other works have been focusing on physics-informed machine learning techniques [27]. Our contributions focus on understanding and representing the complex dynamics through the data-driven lens, and we have worked around deep learning architectures [28] along with specifically designed machine learning models [8, 29, 30]. Our contributions also involve incorporating optimization theory and physical models to enhance the learning performance as well as providing performance guarantees [31, 32, 33, 34].

2.2 Optimization-based Modeling and Control

In this thesis, we also illustrate why optimization can be served as an appropriate language for both system modeling and machine learning. Though these architectures and models have been well studied in many tasks [24, 35, 36, 37], there are still many opportunities in designing specific architectures to achieve either system-level objectives or formally establish performance guarantees. In many cases throughout the thesis, we will start from the unconstrained optimization problem with a parameterized objective

$$\arg\min_{x} \; f_\theta(x), \tag{2.1}$$

while the gradient descent method gives us the basic iterative method for finding better solutions

$$x_{i+1} = x_i - \alpha \Delta_x f_\theta(x). \tag{2.2}$$

On one hand, recent advances in computational framework gives us convenient and efficient tools for auto-differentiation and gradient operation over complex model f_θ [38]. While on the other hand, developing stable and efficient algorithms for many constrained optimization tasks are also necessary for many engineering applications with engineering constraints.

This thesis is also closely related to control theory and reinforcement learning. While typical reinforcement learning methods often try to find a policy

2.3 Power System Operations

In this section, we will start from describing the basic physical models underlying the power grids. We will talk about several control and operation tasks across different timeframes and geographical

scales. Such physical models and system knowledge will shed light on the algorithm design described in later chapters.

2.3.1 Power Flow

we consider a power grid with a tree topology, with a set $\mathcal{N} = \{1, ..., N\}$ of buses and a set $\mathcal{E} \in \mathcal{N} \times \mathcal{N}$ of lines. For each bus i, denote V_i as the voltage magnitude and θ_i as the voltage phase angle; let p_i and q_i denote the active and reactive power injections; let $s_i = p_i + jq_i$ be the complex power injection at bus i. The corresponding active and reactive power injection vectors are denoted as $\mathbf{p} = \begin{bmatrix} p_1 & p_2 & \cdots & p_N \end{bmatrix}^T, \mathbf{q} = \begin{bmatrix} q_1 & q_2 & \cdots & q_N \end{bmatrix}^T$. For each line $(i,k) \in \mathcal{E}$, denote line admittance $y_{ik} = g_{ik} - jb_{ik}$ with $b_{ik} > 0,\ g_{ik} > 0$.

For each bus $i \in \mathcal{N}$, its power injection is governed by

$$p_i = \sum_{k=1}^{N} V_i V_k \left(-g_{ik} \cos\left(\theta_i - \theta_k\right) + b_{ik} \sin\left(\theta_i - \theta_k\right)\right) \tag{2.3a}$$

$$q_i = \sum_{k=1}^{N} V_i V_k \left(g_{ik} \sin\left(\theta_i - \theta_k\right) + b_{ik} \cos\left(\theta_i - \theta_k\right)\right). \tag{2.3b}$$

The power flow equations can also be stated in many forms, and in this thesis, we use the DistFlow formulation [39] most frequently for the ease of comparison with SOCP relaxations and linear approximations. In this formulation, let $s_{ik} = p_{ik} + jq_{ik}$ denote the complex power flow from bus i to bus k and $z_{ik} = 1/y_{ik} = r_{ik} + jx_{ik}$ denote the line impedance. The power flow equations are

$$-p_k = p_{ik} - r_{ik} l_{ik} - \sum_{l:(k,l)\in\mathcal{E}} p_{kl} \tag{2.4a}$$

$$-q_k = p_{ik} - x_{ik} l_{ik} - \sum_{l:(k,l)\in\mathcal{E}} q_{kl} \tag{2.4b}$$

$$V_k^2 = V_i^2 - 2\left(r_{ik} p_{ik} + x_{ik} q_{ik}\right) + \left(r_{ik}^2 + x_{ik}^2\right) l_{ik} \tag{2.4c}$$

$$l_{ik} = \frac{p_{ik}^2 + q_{ik}^2}{V_i^2}. \tag{2.4d}$$

2.3.2 Network Flow Problem

Before going into details about power system optimization and control, we will now take a detour to introduce the general form of network flow problem. A common type of network flow problems arising in logistics and communication network concerns the distribution of goods from origins (e.g., a manufacturing plant) to destinations (e.g., consumers) [33, 40]. Given a graph $G = (V,E)$, the sources, destinations, and intermediate points are collectively modeled as nodes in the set $V = \{1,...,n\}$. The sources have certain amounts of goods and each of the sink has some demand that needs to be satisfied. The goods need not be sent directly from source to destination and may be routed through intermediary points. For each node, let x_i denote its production and let l_i denote the load. We allow a node to have both positive x_i and l_i (a node can both produce and consume goods), so $\mathbf{x} = (x_1,\ldots,x_n)$ and $\boldsymbol{\ell} = (l_1,\ldots,l_n)$ are in $\mathbb{R}^n$. Each of the edge in the network carries some flow of goods and they are related to the nodes through a conservation equation: the sum of the flows into a node must equal to its net load. Algebraically, suppose there are m edges, each with flow f_j. Then the flows $\mathbf{f} = (f_1,\ldots,f_m)$ are related linearly to $\mathbf{x}$ and $\boldsymbol{\ell}$ through an incidence matrix $\mathbf{A}$ of the graph G, where $\mathbf{x} + \mathbf{A}\mathbf{f} = \boldsymbol{\ell}$ and we follow the convention that flows into a node is positive.

We associate a positive c_i with the i'th node, interpreted as the unit cost of producing the good. The standard network flow problem is then to minimize the total costs of production:

$$\min_{\mathbf{x},\mathbf{f}} \quad \mathbf{c}^T\mathbf{x} \tag{2.5a}$$

$$\text{s.t.} \quad \mathbf{0} \le \mathbf{x} \le \bar{\mathbf{x}} \tag{2.5b}$$

$$-\underline{\mathbf{f}} \le \mathbf{f} \le \bar{\mathbf{f}} \tag{2.5c}$$

$$\mathbf{x} + \mathbf{A}\mathbf{f} = \boldsymbol{\ell} \tag{2.5d}$$

where $\mathbf{x}$ and $\mathbf{f}$ are the optimization variables with upper and lower bounds as shown in (2.5b) and (2.5c), respectively. Note, if a node is a pure sink, we can set $\bar{x}_i = 0$.

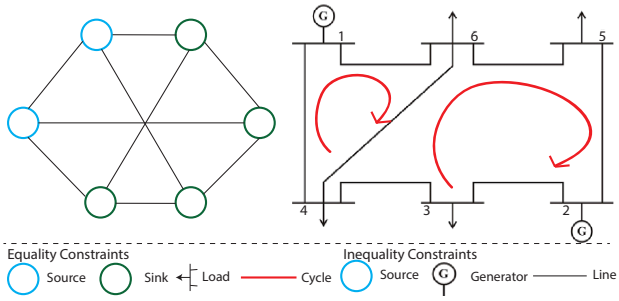

Figure 2.1: The network flow model and OPF model along with physical constraints. Note that cycle equality constraints exist in OPF model due to the underlying power flow equations.

2.3.3 Optimal Power Flow

A network flow problem that includes constraints induced by physical laws is the optimal power flow (OPF) problem. It is one of the most important problems in power system operations, which finds the generations that minimizing cost while satisfying all the loads and flow constraints. The key difference with the problem in (2.5) is that the flows in a power system cannot be set independently. Namely, they must obey *Kirchoff's voltage laws*, which states that a linear combination of the flows must sum up to zero in a cycle in the network as shown in Fig. 2.1.

In the DC power flow model the power flow on the lines are determined by the angle differences. Let θ_i be the angle of bus i and $f_{ij} = b_{ij}(\theta_i - \theta_j)$ be the flow along the line connecting i and j. If a network has cycles, let buses $1,,\ldots,n_c$ be the buses in a cycle, counted in either the clockwise or counterclockwise direction. Consider the weighted sum

$$\begin{aligned}&\frac{f_{12}}{b_{12}} + \frac{f_{23}}{b_{23}} + \cdots + \frac{f_{n_c 1}}{b_{n_c 1}}\\ =&(\theta_1 - \theta_2) + (\theta_2 - \theta_3) + \cdots + (\theta_{n_c} - \theta_1)\\ =&0.\end{aligned}$$

Therefore, the flows lie in a subspace and are not independent from each other. Repeating the above calculation for every cycle gives that the flows lie in a subspace of dimension $n-1$ for a connected network with n buses. A basis of this subspace is called a set of fundamental flows. A basis can be constructed by choosing a spanning tree and consider the flows on the branches as fundamental, and everything else can be derived from these flows.

Figure 2.2 shows an example. Assuming that all the line susceptances are 1 p.u., i.e., $b_{ij} = 1$, and

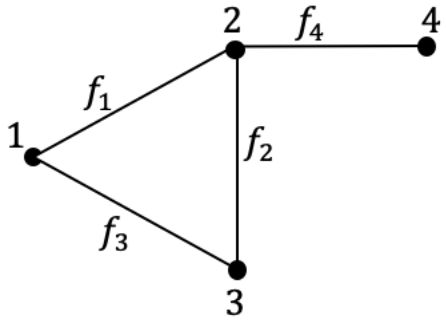

Figure 2.2: Example of DC power flow model.

taking the line flows f_1, f_2, f_4 to the fundamental flows. Then the matrix $\mathbf{K}$ mapping the fundamental flows to all line flows and the matrix $\tilde{\mathbf{A}}$ mapping flows to bus injections are:

$$\mathbf{K} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \tilde{\mathbf{A}} = \begin{bmatrix} 2 & 1 & 0 \\ -1 & 1 & 1 \\ -1 & -2 & 0 \\ 0 & 0 & -1 \end{bmatrix}.$$

Because of these *cycle constraints*, the edge flows must lie in a subspace in $\mathbb{R}^m$. This is the space of fundamental flows, with a dimension of $n-1$. The rest of the $m-n+1$ flows are uniquely determined by the cycle constraints. By selecting a spanning tree of the network where flows on the edges in the tree are fundamental, the rest of the flows on the edges are determined through the cycle constraints [41]. Algebraically, we use $\mathbf{f}$ to denote the flows in a cycle basis and a matrix $\mathbf{K} \in \mathbb{R}^{m\times(n-1)}$ to map it to all of the flows. Then the optimization problem becomes:

$$J(\boldsymbol{\ell}) = \min_{\mathbf{x},\mathbf{f}} \quad \sum_{i=1}^{n} \frac{q_i}{2} x_i^2 + c_i x_i \tag{2.6a}$$

$$\text{s.t.} \quad \mathbf{0} \le \mathbf{x} \le \bar{\mathbf{x}} \tag{2.6b}$$

$$-\bar{\mathbf{f}} \le \mathbf{K}\mathbf{f} \le \bar{\mathbf{f}} \tag{2.6c}$$

$$\mathbf{x} + \tilde{\mathbf{A}}\mathbf{f} = \boldsymbol{\ell}, \tag{2.6d}$$

where $\tilde{\mathbf{A}}$ is the modified incidence matrix mapping the fundamental flows to the nodal injections.

2.3.4 Voltage Regulation

In the distribution grid, other than the standard OPF problem, voltage control is another important problem due to the increasing penetration of renewables and the resulting fluctuations on nodal voltage. The optimal voltage regulation problem is then to optimize system performance subject to the power flow equations (2.4), while the inverter-based distributed resources are available for providing reactive power injections. There are a variety of objective functions and constraints we can consider [42], and throughout this thesis we adopt the following optimal voltage regulation formulation:

$$\min_{\mathbf{q}} \quad \sum_{i=1}^{N} \alpha_i |V_i - V_{i,0}| \tag{2.7a}$$

$$\text{s.t.} \quad \underline{\mathbf{q}} \leq \mathbf{q} \leq \overline{\mathbf{q}} \tag{2.7b}$$

$$\text{Power Flow Equations (2.4)} \tag{2.7c}$$

where α_i is a weight parameter. The goal of the problem is to maintain voltage magnitude V_i within a small distance from the nominal value $V_{i,0}$ for all buses (e.g., plus/minus 5%) [43]. The constraints on reactive powers comes from the rating of the power electronics on the DERs. Note that if a bus does not have reactive power capability, we can simply set the upper and lower bound to be equal to the nominal reactive load value. Here we assume that the active power injections are fixed exogenously (e.g., by solar irradiation), although we can accommodate active power as a optimization variable using the same methodology presented in this paper. In addition, other cost functions such as losses or costs or reactive power can be added to the objective as well.

The optimization problem in (2.7) is not convex because of the quadratic equality (2.4d) in the power flow equations. A simple relaxation is to make it into an inequality as:

$$l_{ik} \geq \frac{p_{ik}^2 + q_{ik}^2}{V_i^2}. \tag{2.8}$$

Using (2.8) instead of (2.4d) in (2.7) makes it a convex optimization problem, in particular a second order cone problem (SOCP) [44, 45].

In Chapter 4 where we will take voltage regulation as a working example for our proposed control algorithm, we will build our work upon an important result which shows the above convex

relaxation is tight under many settings. In fact, there are different convex relaxations one can use, including semidefinite programming (SDP) and the result extends to unbalanced systems as well [46, 47]. Therefore, in principle, there exist efficient algorithms that are guaranteed to find the global optimal solution. However, as we discuss in the next section, these algorithms have rarely been implemented in practice [48].

Chapter 3

UNCERTAINTY MODELING VIA SCENARIO GENERATION

3.1 Motivation

High levels of renewables penetration pose challenges in the operation, scheduling, and planning of power systems. Since renewables are intermittent and stochastic, accurately modeling the uncertainties in them is key to overcoming these challenges [49, 50]. One widely used approach to capture the uncertainties in renewable resources is by using a set of time-series *scenarios* [51]. By using a set of possible power generation scenarios, renewables producers and system operators are able to make decisions that take uncertainties into account, such as stochastic economic dispatch/unit commitment, optimal operation of wind and storage systems, and trading strategies (e.g., see [52, 53, 54, 55] and the references within).

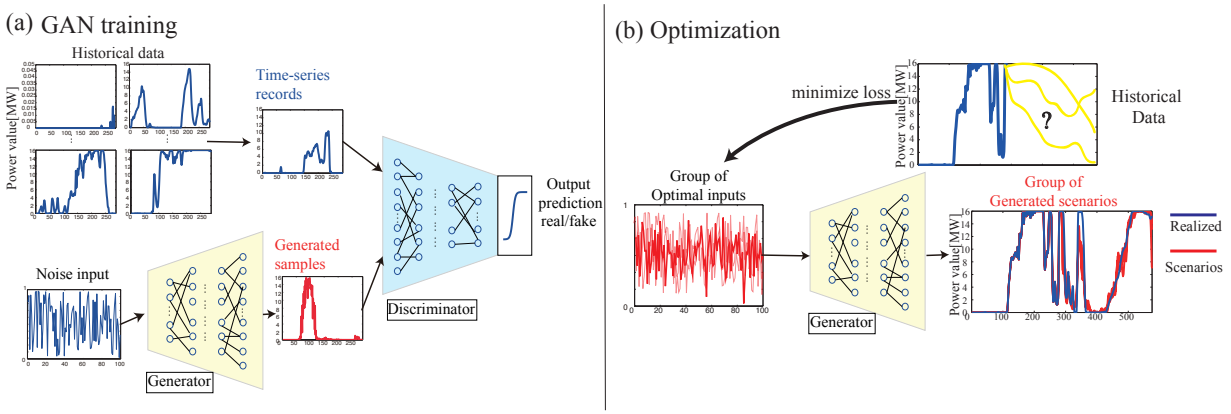

Figure 3.1: Model architecture for our proposed method. During the training process (a), the generator transforms noise vectors into generated time-series, and the discriminator is fed with both historical time-series and generated time-series, and try to discriminate the source of the input. (b). Once training completes, we are able to optimize over the noise vectors to find the future scenarios from generator output conditioned on historical observations.

Despite the tremendous advances recently achieved, scenario generation remains a challenging problem [56, 57]. The dynamic and time-varying nature of weather, the nonlinear and bounded power conversion processes, and the complex spatial and temporal interactions make model-based approaches difficult to apply and hard to scale, especially when multiple renewable power plants are considered. These models are typically constructed based on statistical assumptions that may not hold or difficult to test in practice, and sampling from high-dimensional distributions (e.g. non-Gaussian) is also nontrivial [51]. Previous statistical or physical methods like first-order autoregressive model [58], ensemble methods and Gaussian Copula [1, 52, 59] either required strong statistical assumptions or detailed physical measurements and modeling. What is more, most of these methods focus on capturing the marginal distribution of each individual time slots of the forecasting horizon, while paying less attention to the temporal correlations in the scenarios [60]. In addition, some of these methods depend on certain probabilistic forecasts as inputs, which may limit the diversity of the generated scenarios and under-explore the overall variability of renewable resources.

Specifically, we propose to utilize the power of the recently discovered machine learning concept of *Generative Adversarial Networks* (GAN) [30] to fulfill the task of scenario generation. Generative models have become a research frontier in computer vision and machine learning area, with the promise of utilizing large volumes of unlabeled training data. There are two key benefits of applying such class of methods. The first is that they can directly generate new scenarios based on historical data, without explicitly specifying a model or fitting probability distributions. The second is that they use unsupervised learning, avoiding cumbersome manual labellings that are sometimes impossible for large datasets. In the image processing community, GANs are able to generate realistic images that are of far better quality compared to other methods [30, 61, 62].

The intuition behind GAN is to leverage the power of deep neural networks (DNNs) to both express complex nonlinear relationships (the generator) as well as classify complex signals (the discriminator). The key insight of GAN is to set up a minimax two player game between the generator DNN and the discriminator DNN (thus the use of "adversarial" in the name). During each training epoch, the generator updates its weights to generate "fake" samples trying to "fool" the

discriminator network, while the discriminator tries to tell the difference between true historical samples and generated samples. In theory, at reaching the Nash equilibrium, the optimal solution of GANs will provide us a generator that can exactly recover the distribution of the real data so that the discriminator would be unable to tell whether a sample came from the generator or from the historical training data. At this point, generated scenarios are indistinguishable from real historical data, and are thus as realistic as possible. Fig. 3.1(a) shows the general architecture of a GANs' training procedure under our specific setting. Our approach can be used for a variety of scenario forecasts problems, e.g., wind and solar generation, and is easy to adjust the forecast horizon (e.g. ranging from from 1-2 hours to 1-2 days) with little tuning. Specifically, we make the following contributions:

To extend the pure scenario generation, we are also interested in forecasting a group of scenarios which could inform system operators the possible future realizations of power generation process. That is, given a time-series of renewables generation realization, we are interested to formulate and solve an optimization problem iteratively to obtain high-quality scenario forecasts. Such optimization setup and solution procedures are illustrated in Fig. 3.1(b).

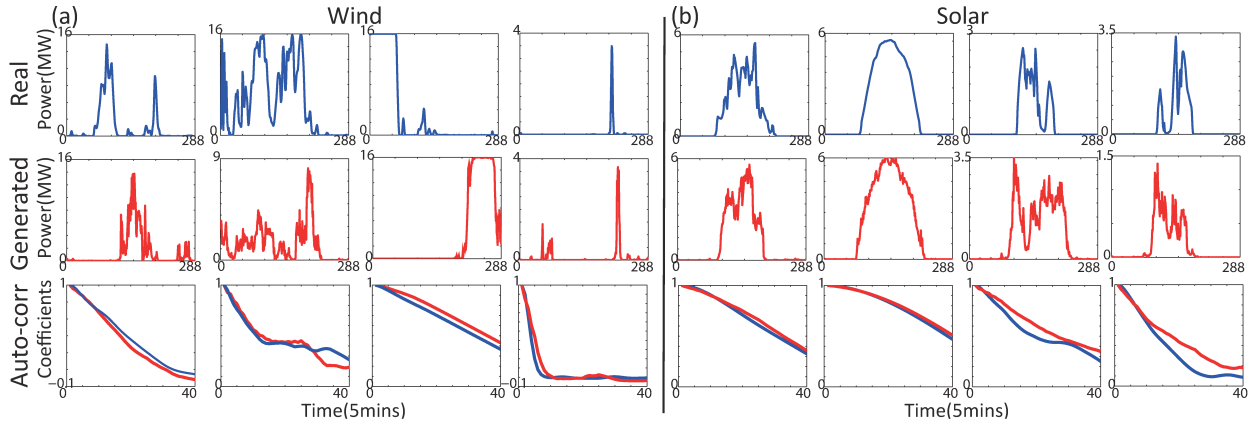

Figure 3.2: Selected samples from our validation sets (top) versus generated samples from our trained GANs (middle) for both wind and solar groups. The pair of samples are selected using Euclidean distance based search. Without using these validation samples in training, our GANs is able to generate samples with similar behaviors and exhibit a diverse range of patterns. The autocorrelation plots (bottom) also verify generated samples' ability to capture the correct time-correlations.

3.2 Problem Setup

In this section, we describe the setup for GANs [30] and how we could adapt it to the task of scenario generation and scenario forecasts. We first formulate the training objectives for the discriminator and the generator respectively, and show GANs is a good fit to generate a potentially unlimited number of renewable power production time-series. Then we illustrate how this time-series producer can be served in an optimization problem to find desired scenarios forecasts.

Denote observations x_j^t for times $t \in T$ of renewable power production are available for each power plant j, $j = 1,...,N$. We also denote the true distribution of the observation as $\mathbb{P}_x$, which is unknown and maybe difficult to model because of complex spatial and temporal correlations. Suppose we have access to a group of noise vector input z under a known distribution $z \sim \mathbb{P}_z$ that is easily sampled from (e.g., jointly Gaussian or uniform). Given a sample z drawn from p_z, our goal is to find a function G such that after transformation, $G(z)$ follows $\mathbb{P}_x$. This is accomplished by simultaneously training two deep neural networks: the generator network $G(z;\theta^{(G)})$ and the discriminator network $D(x;\theta^{(D)})$. Here, $\theta^{(G)}$ and $\theta^{(D)}$ denote the weights of two neural networks, respectively. For convenience, we sometimes suppress the symbol θ.

Generator: During the training process, the generator is trained to take a batch of inputs from the noisy distribution $\mathbb{P}_z$, and by taking a series of up-sampling operations by neurons of different functions, and to output realistic time-series samples. Ideally, they should appear as if drawn from $\mathbb{P}_x$. Therefore, after training finishes, the mapping $G(z;\theta^{(G)})$ should follow the true data distribution $\mathbb{P}_x$.

Discriminator: The discriminator is trained simultaneously with the generator. It takes input samples either coming from real historical data or coming from the generator. By taking a series of operations of down-sampling using another deep neural network, it outputs a continuous value p_{real} that measures to what extent the input samples belong to $\mathbb{P}_x$. The discriminator can be expressed as $D(x;\theta^{(D)})$, where x may come from $\mathbb{P}_x$ or $\mathbb{P}_z$. The discriminator is trained to learn to distinguish between $\mathbb{P}_x$ from $\mathbb{P}_z$, and thus to maximize the difference between $D(x)$ (x from real data) and $D(G(z))$.

With the objectives for the discriminator and the generator defined, we can now formulate loss function L_G for the generator and L_D for the discriminator to train to optimize the performance of them (i.e., update neural networks' weights based on the losses). A small L_G reflects that $G(z)$ is as realistic as possible from the discriminator's perspective, e.g., the generated scenarios are "looking like" historical scenarios to the discriminator. Similarly, a small L_D indicates discriminator is good at telling the difference between generated scenarios and historical scenarios, which means there is a large difference between $\mathbb{P}_{G(z)}$ and $\mathbb{P}_x$. Following this guideline and the loss defined in [63], we define L_D and L_G as:

$$L_G = -\mathbb{E}_{z\sim\mathbb{P}_z}[D(G(z))] \tag{3.1a}$$

$$L_D = -\mathbb{E}_{x\sim\mathbb{P}_x}[D(x)] + \mathbb{E}_{z\sim\mathbb{P}_z}[D(G(z))]. \tag{3.1b}$$

In the above, the expectations are taken as empirical averages based either on the historical data or on the generated data. Note the functions D and G are parametrized by the weights of two distinct deep neural networks.

We can now combine (3.1a) and (3.1b) to construct the minimax game value function $V(G,D)$ for these two players:

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{data}(x)}[D(x)] - \mathbb{E}_{z\sim\mathbb{P}_z}[D(G(z))] \tag{3.2}$$

where $V(G,D)$ is the negative of L_D.

During first few training iterations, G just generates time-series samples $G(z)$ totally different from samples in $\mathbb{P}_x$, and after learning from those samples coming from $\mathbb{P}_x$, the discriminator is able to reject $G(z)$ with high confidence. In that case, L_D is small, and L_G, $V(G,D)$ are both large. The generator gradually learns to generate more realistically looking samples, while at the same time the discriminator is also trained to distinguish these newly fed generated samples from G. As training moves on and moves close the the equilibrium, G is able to generate samples that look as realistic as real power generation time-series corresponding to a small L_G value, while D is unable to distinguish $G(z)$ from $\mathbb{P}_x$ with large L_D. Eventually, we are able to learn an unsupervised representation of the probability distribution of renewables time-series. By sampling $z \sim \mathbb{P}_z$, we get $G(z)$ that appears "as if" it was sampled from the true distribution.

More formally, the minimax objective (3.2) of the game can be interpreted as the dual of the so-called Wasserstein distance (Earth-Mover distance) [64]. The Wasserstein distance between two distribution $\mathbb{X}$ and $\mathbb{Y}$ measures the effort (or "cost") needed to transport $\mathbb{X}$ to $\mathbb{Y}$. It is shown in [63] that we are precisely trying to get two distributions, $\mathbb{P}_x$ and $\mathbb{P}_{G(z)}$ to be close to each other by defined loss for G and D in (3.1a) and (3.1b) respectively.

Note that unlike previous approaches for generating scenarios given historical observations, which all involve the modeling of renewables generation stochastic processes [1, 52, 59], by using GANs we bypass the step of learning or modeling $\mathbb{P}_x$ explicitly.

As is shown in Fig. 3.3b, once the Wasserstein distance we estimate using 3.2 stops decreasing or reaches pre-set limits, the "cost" of transforming a generated sample to original sample has been minimized. So the distance between the distribution of $G(Z)$ and $\mathbb{P}_X$ is minimized. Thus we find the optimal generator G^*. In the GANs community, there is a growing body of literature about the choice of loss functions. Here we chose to use the Wasserstein distance [63] instead of the original Jensen-Shannon divergence proposed in [30]. This is because Wasserstein distance directly calculates the distance between two distributions $\mathbb{P}_G$ and $\mathbb{P}_X$. Since we want to generate scenarios that reflect the variability of renewables generation processes, training using Wasserstein distance allows us to capture all of the modes in training samples which are all coming from $\mathbb{P}_X$. Training using Jensen-Shannon divergence tends to lead the generator to generate a single pattern of power profile that has the highest probability. In this section, we want to generate scenarios that reflect diverse modes of renewables, which is accomplished by using the Wasserstein distance to directly measure the distance between distributions of real historical data and generated samples.

In an unconditioned generative model, we do not control the specific types of the samples being generated by G. Sometimes, we are interested in scenarios "conditioned on" certain class of events, e.g., calm days with intermittent wind, or windy days with farms at full load capacity. Conditional generation is done by incorporating more information to the training procedure of GANs, such that the generated samples conforming to same properties as certain class of training samples. Inspired by supervised learning where we have labels for each training samples, here we propose to combine event labels with training samples, and thus the objective for G is to generate samples under given

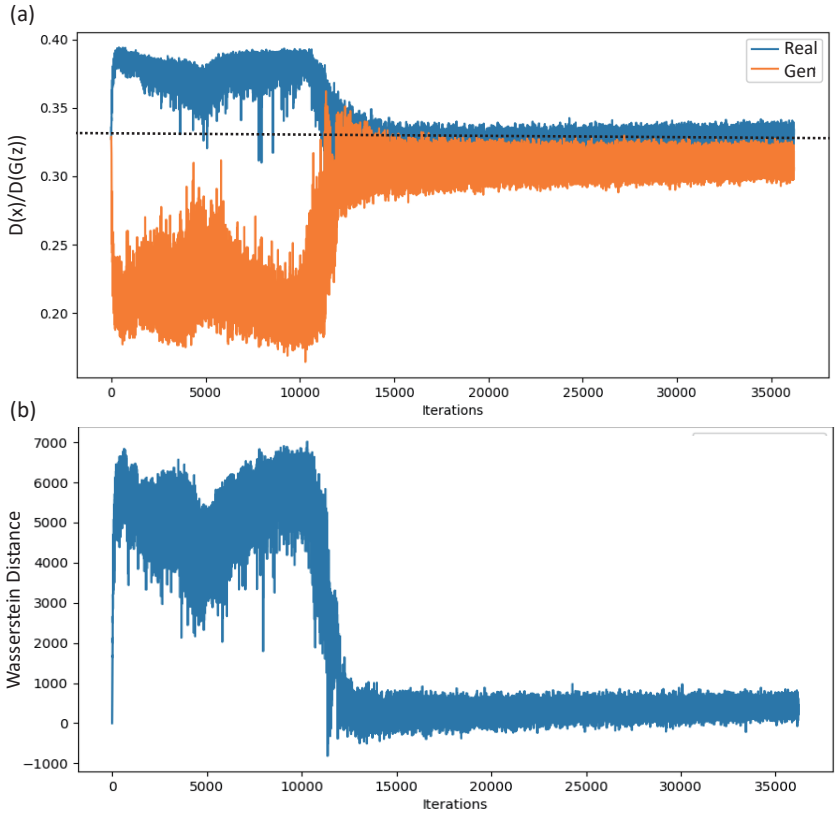

Figure 3.3: Training evolution for GANs on a wind dataset. (a) The outputs from the discriminator $D(x)/D(G(z))$ during training illustrates the evolution of generated samples $G(z)$. At the start, the generated samples (orange) and the real samples (blue) are easily distinguished at the discriminator. As training progresses, they are increasing difficult to distinguish. (b) The empirical Wasserstein distance between the distribution of the real sample and the generated samples, where close to zero means that the two distributions are close to each other.

class [61]. More formally, the problem can be written as:

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{data}(x)}[D(x|y)] - \mathbb{E}_{z\sim \mathbb{P}_z}[D(G(z|y))], \tag{3.3}$$

where y encodes different type of classes of conditions.

Class labels are assigned based on user-defined classification metrics, such as the mean of daily power generation values, the month of training samples coming from, etc. Note that such class labels are just representation of samples' events reflected by the power generation data distribution, GANs should be able to learn the conditional distribution and generate samples based on any given

meaningful classification metric. Since such conditional GANs is only modifying the unconditional GANs model proposed earlier with a label vector input, both models can be trained using similar algorithm described in Algorithm 1.

Algorithm 1 GANs for Time-Series Generation

Require: Learning rate η, clipping parameter c, batch size m, Number of iterations for discriminator per generator iteration n_{discri}

Ensure: Initial weights $\theta^{(D)}$, $\theta^{(G)}$

while $\theta^{(D)}$ has not converged **do**

for $t = 0, ..., n_{discri}$ **do**

 # Update parameter for Discriminator

Sample batch from historical data:

$\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_x$

Sample batch from Uniform distribution:

$\{z^{(i)}\}_{i=1}^{m} \sim Unif(-1,1)$

Update discriminator nets using gradient descent:

$g_{\theta^{(D)}} \leftarrow \nabla_{\theta^{(D)}}[-\frac{1}{m}\sum_{i=1}^{m} D(x^{(i)}) + \frac{1}{m}\sum_{i=1}^{m} D(G(z^{(i)}))]$

$\theta^{(D)} \leftarrow \theta^{(D)} - \eta \cdot RMSProp(\theta^{(D)}, g_{\theta^{(D)}})$

$\theta^{(D)} \leftarrow clip(\theta^{(D)}, -c, c)$

end for

 # Update parameter for Generator

Update generator nets using gradient descent:

$g_{\theta^{(G)}} \leftarrow \nabla_{\theta^{(G)}} \frac{1}{m}\sum_{i=1}^{m} D(G(z^{(i)}))$

$\theta^{(G)} \leftarrow \theta^{(G)} - \eta \cdot RMSProp(\theta^{(G)}, g_{\theta^{(G)}})$

end while

Algorithm 2 Forecasting Scenarios with GANs

Require: Pre-trained GANs model weights $\theta^{(G)}$, $\theta^{(D)}$

Require: Measurements $\mathbf{p}_{hist}$, point forecast $\hat{\mathbf{p}}_{pred}$

Require: PI level α, initial PI level α_{sub}, weighting parameters γ, β, initial point finder iterations n_{init}, scenario finder iterations n_{scen}, learning rate η, scenario number N

Ensure: Generated scenarios S $\leftarrow \emptyset$

for $scenario = 0, ..., N$ **do**

Sample $\mathbf{p}_{initial} \sim Unif(L^{\alpha_{sub}}(\hat{\mathbf{p}}_{pred}), U^{\alpha_{sub}}(\hat{\mathbf{p}}_{pred}))$

Sample $z \sim Unif(-1, 1)$

 # Find good initial z

for $iteration = 0, ..., n_{init}$ **do**

Update z using gradient descent:

$g_z \leftarrow \nabla_z L_{sub}$ *#L_{sub} is defined by 3.8*

$z \leftarrow z - z \cdot MomentumGD(z, g_z)$

$z \leftarrow clip(z, -1, 1)$

end for

 # Find forecasting scenarios

for $iteration = 0, ..., n_{scen}$ **do**

Update z using gradient descent:

$g_z \leftarrow \nabla_z L_{main}$ *#L_{main} is defined by 3.7*

$z \leftarrow \theta^{(D)} - \eta \cdot MomentumGD(z, g_z)$

$z \leftarrow clip(z, -1, 1)$

end for

S.insert(G(z))

end for

3.3 Applications of Scenario Generations

3.3.1 Single Time-Series Scenario Generation

Consider a set of historical data for a group of renewable resources at N sites. For site j, let $\mathbf{x}_j$ be the vector of historical data indexed by time, $t = 1, \ldots, T$, and j ranges from 1 to N. Our objective is to train a generative model based on GANs by utilizing historical power generation data $\{\mathbf{x}_j\}, j = 1, \ldots, N$ as the training set. Generated scenarios should be capable of describing the same stochastic processes as training samples and exhibiting a variety of different modes representing all possible variations and patterns seen during training.

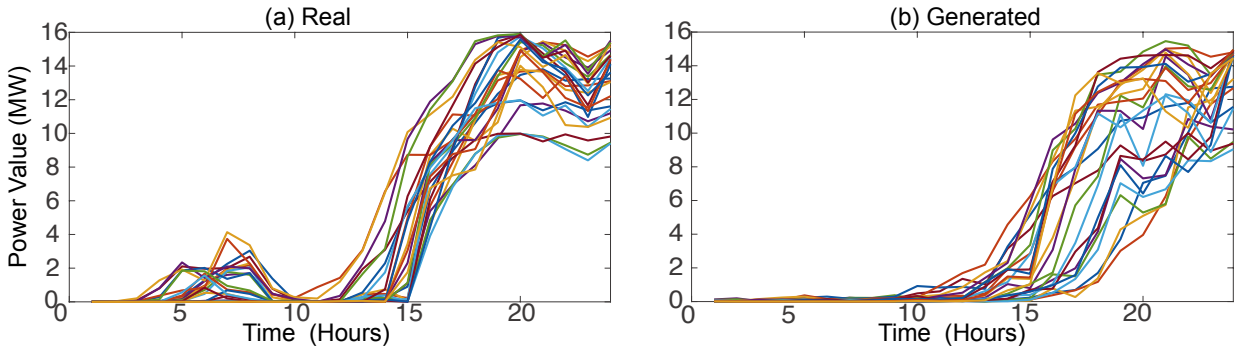

Figure 3.4: A group of one-day historical (top) and generated (bottom) wind farm power output. The latter behaves similarly to the former both spatially and temporally.

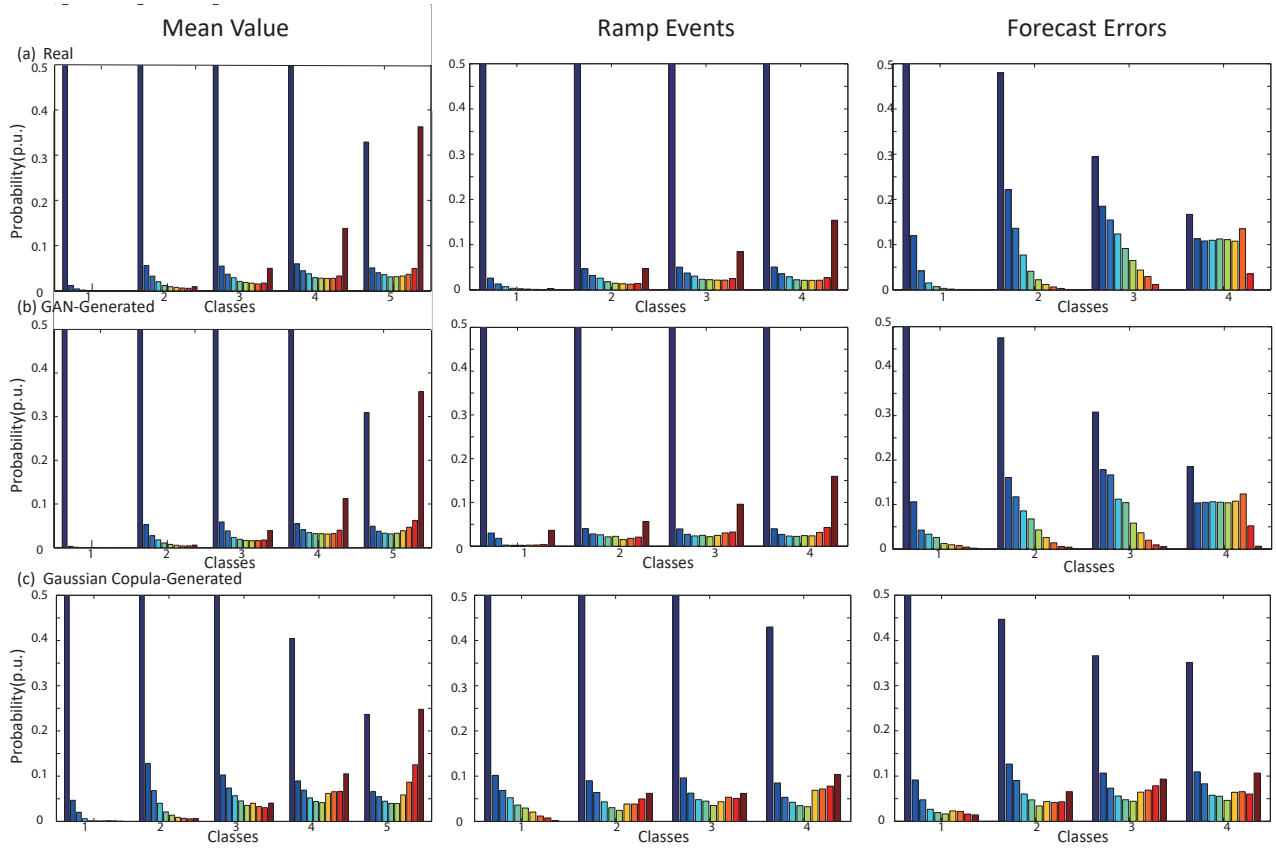

Figure 3.5: Three group of conditional scenario generation for wind power comparing the marginal distribution of the real data, samples generated by GANs, and the samples generated by Gaussian copula. Three conditions, the wind power generations' mean value, 5-min ramp events, forecast errors level, are fed as conditional labels into our proposed model (Fig. 3.5b), and the generated scenarios' marginal distribution is compared with both the validation data (Fig. 3.5a) and scenarios generated by Gaussian copula method (Fig. 3.5c). In all cases, the marginal distribution of the samples generated by Gaussian copula tend to be more spread out than the actual marginal distributions and cannot accurately capture the extreme values. In contrast, the marginal distribution of the samples generated by GANs very closely resembles the real data distributions.

3.3.2 Scenario Generation for Multiple Sites

In a large system, multiple renewable resources needs to be considered at the same time. Here we are interested in simultaneously generating multiple scenarios for a given group of geographical close sites. We have historical power generation observations $\{\mathbf{x}_j\}, j = 1,\ldots,N$ for N sites of interests

with the same time horizon. The generated scenarios should capture both the temporal and spatial correlations between the resources, as well as the marginal distribution of each individual resource.

In some situations a point forecast is given and scenarios should be thought as the forecasting error. Our approach can be easily applied by simply replacing the training samples with the historical forecast errors. Based on different forecasting technologies, there may or may not be correlations among the errors. Our approach would automatically generate statistically correct scenarios without any explicit assumptions.

3.3.3 Event-Based Scenario Generation

In addition to the standard scenario generation process described above, we may want to generate scenarios with distinct properties. For instance, an operator may be interested in scenarios that capture the solar output of a hot summer day. We incorporate these given properties into the training process by labeling each training samples with an assigned label to represent the event. Specifically, we use a label vector $\mathbf{y}$ to classify and record certain properties in an observation $\mathbf{x}_j$.

Thus in this part we are interested in scenario generation conditioned on the label y, while samples having same label should follow the similar properties. Our objective here is to train a generative model based on GANs using historical conditional power generation data $\{\mathbf{x}_j|y_j\}, j = 1,\ldots,N$ as a training set.

3.4 Scenario Forecasts Based on Trained GANs

So far we have talked about using GANs as a powerful time-series modeling unit. Yet for many tasks in power system operations, the key information we want to know is about possible future scenarios (e.g., day-ahead scenarios for wind power generation or electricity demand). For a typical renewable power generation site, assume at timestep t, we have records for actual past power outputs $\mathbf{p}_{hist} \in \mathbf{R}^{h+1}$ with $\mathbf{p}_{hist} = [p_t \ \ldots \ p_{t-h}]^T$. Meanwhile, we have some forecasting method to obtain the point forecasts $\hat{p}_{t+i}$ for each look-ahead time $i = 1,...,k$ given $\mathbf{p}_{hist}$. This forecast is denoted by $\hat{\mathbf{p}}_{pred} = [p_{t+1} \ \ldots \ p_{t+k}]^T$, where k is the forecasting horizon. Based on the historical information and

the point forecast, we are interested in generating a group of N scenarios $S = \{s_1, ..., s_N\},\ s_i \in \mathbf{R}^k$, which represent the possible variations around the point forecast and accurately reflect the temporal dynamics of future generation. Note we focus on the scenario forecasting problem, so the central point forecast can be provided by any method.

Assume we have trained a GANs model based on the set of observations. Given some input noise z, $G(z)$ generates a possible realization without regarding to the historically observed data $\mathbf{p}_{hist}$ and the point forecast $\hat{\mathbf{p}}_{pred}$. Therefore, we need to *constrain the possible* $G(z)$ *to satisfy two conditions:* 1) the part of $G(z)$ from time index $t-h$ to t should be close to the historical data $\mathbf{p}_{hist}$; 2) the part of $G(z)$ from time index $t+1$ to $t+k$ should be realistic and respect the point forecast.

To describe these conditions, we introduce two projection operators that separate a vector into two parts. Given $\mathbf{v} \in \mathbf{R}^{h+k+1}$, we denote two projection operations $\mathbb{P}_{hist}(f(\mathbf{v})) = [v_1, ..., v_{h+1}]^T$ and $\mathbb{P}_{pred}(f(\mathbf{v})) = [v_{h+2}, ..., v_{h+k+1}]^T$ to extract former $h+1$ and latter k dimensions of $\mathbf{v}$, respectively.

Meanwhile, we want to constrain generated scenarios do not conflict with the information provided by point forecasts $\hat{\mathbf{p}}_{pred}$ (e.g., information from numerical weather prediction (NWP)). Then we can constrain latter part of $G(z)$ so that they do not conflict with the given $\hat{\mathbf{p}}_{pred}$. Then to ensure that the first part of $G(z)$ resembles $\mathbb{P}_{hist}$, we use the following cost function:

$$||\mathbb{P}_{hist}(G(z)) - \mathbf{p}_{hist}||_2. \tag{3.4}$$

To ensure the generated scenarios are realistic, we add a loss term $-D(G(z))$ where D is the discriminator output (recall larger discriminator ouput indicates more realistic samples). Finally, we use the point forecast $\hat{\mathbf{p}}_{pred}$ by defining a *prediction interval* that the generated scenarios should lie in [65]. We describe this interval with an upper bound $U^{\alpha}(\hat{\mathbf{p}}_{pred})$ and a lower bound $L^{\alpha}(\hat{\mathbf{p}}_{pred})$, controlled by a parameter α (can be interpreted as the prediction confidence):

$$L^{\alpha}(\hat{\mathbf{p}}_{pred}) = \frac{1}{\alpha}\hat{\mathbf{p}}_{pred}, \quad U^{\alpha}(\hat{\mathbf{p}}_{pred}) = \alpha\,\hat{\mathbf{p}}_{pred} \tag{3.5}$$

Using all of the objectives and constraints above, given the observation and forecast vector pair $\mathbf{p}_{hist}$, $\hat{\mathbf{p}}_{pred}$, and GANs pre-trained model G, D, the scenario forecasts problem can be formulated as

a constrained optimization problem:

$$\min_{z} \quad ||\mathbb{P}_{hist}(G(z)) - \mathbf{p}_{hist}||_2 - \gamma D(G(z)) \tag{3.6}$$

$$s.t. \quad z \in \mathcal{Z} \tag{3.6a}$$

$$L^{\alpha}(\hat{\mathbf{p}}_{pred}) \leq \mathbb{P}_{pred}(G(z)) \leq U^{\alpha}(\hat{\mathbf{p}}_{pred}). \tag{3.6b}$$

where γ is a weighting parameter; (3.6a) constrains z to be within the domain of $G(z)$, which we take to be a hypercube $\mathcal{Z} = [-1,1]^{h+k+1}$; (3.6b) constrains the generators' output to be within the given prediction intervals given $\hat{\mathbf{p}}_{pred}$. By solving above optimization problem, we can obtain a forecasting scenario $\mathbb{P}_{pred}(G(z^*))$.

Since both of the objective and constraints in (3.6) are non-convex, to deal with the inequality constraints (3.6b), we propose to substitute it into the main objective with two log barriers. Then the optimization problem is reformulated as

$$\begin{aligned}\min_{z} \quad & ||\mathbb{P}_{hist}(G(z)) - \mathbf{p}_{hist}||_2 - \beta \ \log(\mathbb{P}_{pred}(G(z)) - L^{\alpha}(\hat{\mathbf{p}}_{pred})) - \\ & \beta \ \log(U^{\alpha}(\hat{\mathbf{p}}_{pred}) - \mathbb{P}_{pred}(G(z))) - \gamma D(G(z)) \end{aligned} \tag{3.7a}$$

$$s.t. \quad z \in \mathcal{Z}. \tag{3.7b}$$

where β is the weighting parameter for log barriers.

We then go into the task of finding possible future scenarios based on trained GAN model. Because of the highly non-convex nature of $G(\cdot)$ and $D(\cdot)$, there exist many local optima in (3.6). The key to finding a group of solutions to (3.6) exploits that fact. To ensure that we reach a good local optimum, we add momentum to the gradient descents algorithm [66] to skip from saddle points and shallow local optima.[1]

Since there are multiple local optima to (3.6), we can start at different initial points $z_i \in \mathcal{Z}$ and find distinct forecasting scenario $\mathbb{P}_{pred}(G(z_i^*))$ by solving (3.7) using gradient descents with momentum (MomentumGD). As the training loss defined in (3.2) incurs G to generate diverse

[1]There is a growing body of literature on the local optima of non-convex functions and interested readers can refer to [66] and the references within.

modes given different z, we are able to obtain a group of distinct yet realistic scenarios with different initial starting values.

In order to obtain good starting points for z which $\mathbb{P}_{pred}(G(z)))$ do not fall outside of the log barriers in (3.7), we first solve the following subsidiary problem:

$$\min_{z} \quad ||\mathbb{P}_{pred}(G(z)) - \mathbf{p}_{initial}||_2 \tag{3.8}$$

$$s.t. \quad z \in \mathcal{Z} \tag{3.8a}$$

$$L^{\alpha}(\hat{\mathbf{p}}_{pred}) \leq \mathbf{p}_{initial} \leq U^{\alpha}(\hat{\mathbf{p}}_{pred}). \tag{3.8b}$$

where $\mathbf{p}_{initial}$ is sampled uniformly at random from $[L^{\alpha}(\hat{\mathbf{p}}_{pred}),\ U^{\alpha}(\hat{\mathbf{p}}_{pred})]$. In order that we can always obtain a good initial z, we set α in (3.8) to be slightly smaller than α in main objective function (3.7). In Algorithm 2 we summarize our approach for generating a group of scenarios provided with a pre-trained GANs weights as well as pairing historical measurements and point forecasts.

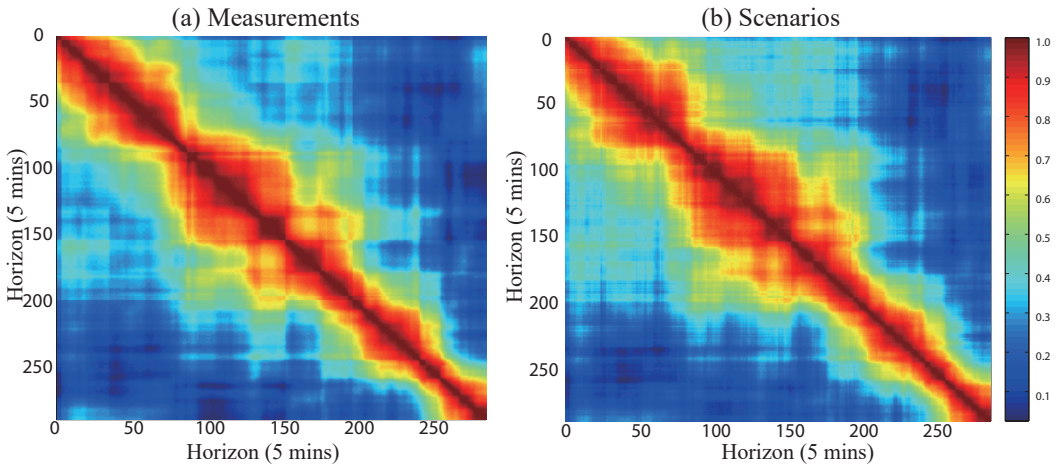

Figure 3.6: The Pearson's correlation matrix for a group of realizations (left) and scenarios forecasts (right) indicates our scenario forecasts capture the linear correlation for different forcasting lead time.

3.5 Experimental Results

3.5.1 Data Collection

We build training and validation dataset using power generation data from NREL Wind[2] and Solar[3] Integration Datasets [67]. The original data has resolution of 5 minutes. We choose 24 wind farms and 32 solar power plants located in the State of Washington to use as the training and validating datasets. We shuffle the daily samples and use 80% of them as the training data, and the remaining 20% as the testing datasets Along with the wind read measurements, we also collect the corresponding 24-hour ahead forecast data, which is later used for conditional generation based on forecasts error. The 10% and 90% quantile forecasts are also available for Gaussian copula method setup. All of these power generation sites are of geographical proximity which exhibit correlated (although not completely similar) stochastic behaviors. Our method can easily handle joint generation of scenarios across multiple locations by using historical data from these locations as inputs with no changes to the algorithm. Thus the spatiotemporal relationships are learned automatically.

Fig. 3.2 shows examples of our generated daily scenarios with a comparison to historical scenarios. These generated scenarios correctly capture the rapid variations and strong diurnal cycles in wind and solar. Note we explicitly chose examples where the historical data and the generated scenarios do not match each other perfectly. Our goal is to generate *new and distinct* scenarios that capture the intrinsic features of the historical data, but not to simply memorize the training data. Moreover, in the task of scenario forecasts, we are interested in generating all possible scenarios to capture possible temporal patterns in the future renewables generations.

3.5.2 Spatio-Temporal Correlation

Instead of feeding a batch of sample vectors $x^{(i)}$ representing a single site's diurnal generation profile, here we feed GANs with a real data matrix $\{x^{(i)}\}$ of size $N \times T$, where N denotes the

[2]https://www.nrel.gov/grid/wind-integration-data.html

[3]https://www.nrel.gov/grid/sind-toolkit.html

total number of generation sites, while T denotes the total number of timesteps for each scenario. Here we choose $N = 24, T = 24$ with a resolution of 1 hour. A group of real scenarios $\{x^{(i)}\}$ and generated scenarios $\{G(z^{(i)})\}$ for the 24 wind farms are ploted in Fig. 3.4. By visual inspection we find that the generated scenarios retain both the spatial and temporal correlations in the historical data (again, not seen in the training stage).

3.5.3 Conditional Generation

In this setting we are adding class information for the GANs' training procedure, and the generated samples could be "conditioned" to certain context or statistics informed by such labels. Here we present four representative applications using class information. For all these practical conditional scenario generation applications, we observe the generated samples are realistic. And we illustrate in Fig. 3.5 the learned overall marginal distribution compared to validation set in three different tasks with different settings of labels: mean value, ramp events and forecast errors.

3.5.4 Scenario Forecasts

We make use of the Pearson's correlation coefficient, which is a standard method to evaluate the linear relationship of time-series at various look-ahead times. Given the set of generated scenarios or realizations S, each term $\rho_{i,j}$ in the Pearson's correlation matrix denotes the Pearson correlation for lead time i and j, and is calculated by

$$\rho_{i,j} = \frac{Cov(S_i, S_j)}{\sigma_{S_i}\sigma_{S_j}} \tag{3.9}$$

where $Cov(S_i, S_j)$ is the covariance of S_i and S_j.

We validate if forecasted scenarios coming from our method have similar temporal correlation as the actual wind power values. In Fig. 3.6 we plot the colormap for the covariance matrix of a group of 32 wind turbines' $24-$hour actual measurements, along with $1,600$ forecasting scenarios with 50 scenarios for each realization. The $x-$ and $y-$ axes are for the prediction horizon k. Similar covariance matrix element values indicate that without any model assumptions being made

during training process, our proposed scenario generation method is able to capture the temporal dependency accurately.

In Fig. 3.7 we specifically select one 24−hour sample whose point forecast is deviating a lot from the actual measurements. By selecting different prediction intervals, our proposed method could reflect the trade-off between reliability and sharpness. When the interval level is $\alpha = 1.5$, generated scenarios are close to point forecasts, yet fail to cover the realizations; while when $\alpha = 3$, generated scenarios could cover the actual power production values, but are less concentrated.

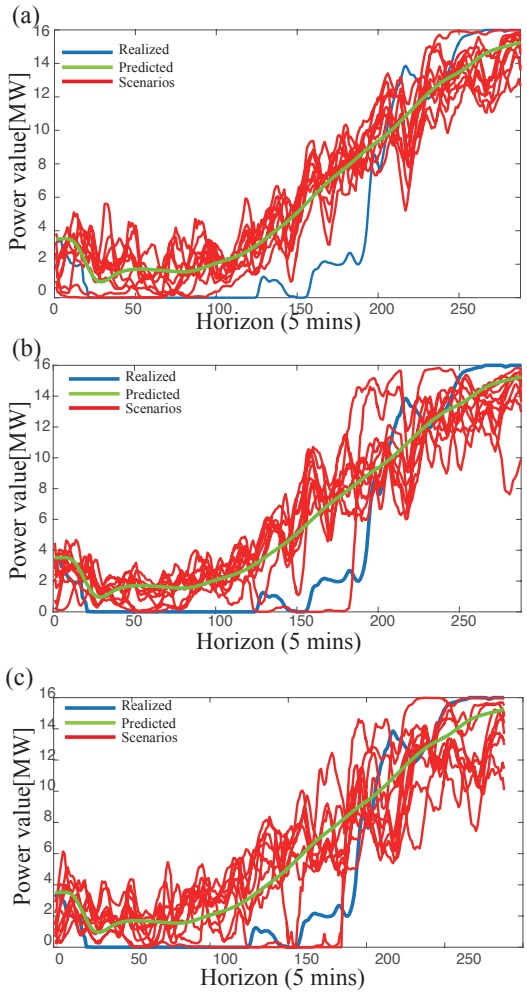

Figure 3.7: Plot (a) (b) (c) correspond to group of 10 day-ahead scenarios (red) with varying PIs of 1.5, 2 and 3 respectively.

In order to verify the group of generated scenarios are able to represent possible future realizations, the scenarios should be able to cover the actual value of power generation (reliable), while at the same time distance between generated scenarios should be small (sharp). We make use of the Continuous Ranked Probability Score (CPRS) [68], which is a negatively-oriented score (smaller scores are better) that jointly evaluates the reliability and sharpness of generated scenarios. The score at lead time k is defined as follows:

$$CPRS_k = \frac{1}{N}\sum_{t=1}^{N}\int_0^1 (\hat{F}_{t+k|t}(p) - I(p \geq p_{t+k}))^2 dp \tag{3.10}$$

where N is the total number of evaluated scenarios, $\hat{F}_{t+k|t}(p)$ is the cumulative distribution for normalized generated scenarios' value at lead time k, and $I(p \geq p_{t+k})$ is the indicator function to compare the normalized scenarios and measurements. Since we are not using quantile statistics to calculate $\hat{F}_{t+k|t}(p)$, we use the discrete-valued $\hat{F}_{t+k|t}(p)$ to calculate (3.10).

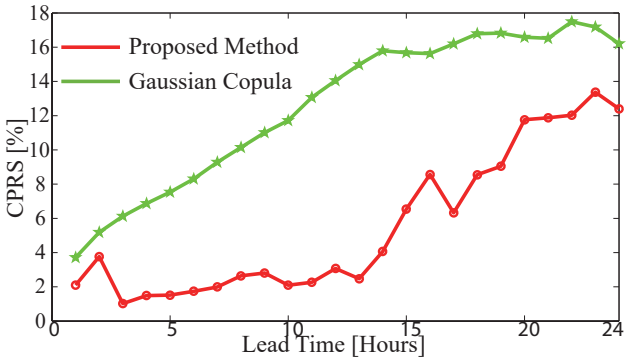

Figure 3.8: CPRS of scenarios generated by proposed method and empirical Gaussian Copula [1].

The performance of the proposed method is also demonstrated by the CPRS score. Results for our approach and Gaussian copula method are plotted in Fig. 3.8. Both approaches use the same training dataset to get the time-series generator or to find the estimate of the covariance matrix, and are tested on the stand-alone testing samples. The proposed method has better performance at different lead time compared to Gaussian Copula. Since point forecasts normally accumulate larger errors with longer forecast horizons, both methods have growing CPRS values with respect to forecasting horizons.

3.6 Conclusion

In this chapter by built upon the novel generative adversarial networks, we proposed a data-driven unsupervised machine learning approach for forecasting scenarios of renewables power generation processes. The proposed method is flexible and easily implemented in problems with high penetration of renewables. Numerical results show that comparing with existing scenario generation approaches, the proposed method is able to generate realistic, high quality scenarios capturing spatiotemporal behaviors of renewables without any explicit model construction.

Scenario approach to address the data uncertainty have been proved to be a viable route for many decision-making and control problems. In the next two chapters we will introduce machine learning methods for energy system operation and control under deterministic environments. However for the future work, it is worthwhile to explore venues where we can combine the uncertainty modeling units and decision-making algorithms organically. Such issues are becoming increasingly important considering the uncertain scenarios brought by climate change coupled with demand-side revolution. What is the worst case guarantee for system performance under extreme weather? What are the all possible electricity demand profiles considering higher penetration of EVs and behind-the-meter distributed generations? Can we learn to operate the grid by navigating through the extreme scenarios? Joint developments of machine learning models and rigorous system-level analysis will be conducted in the future work.

Chapter 4

OPTIMAL CONTROL VIA NEURAL NETWORKS

4.1 Introduction

Decisions on how to best operate and control complex physical systems such as the power grid, commercial and industrial buildings, transportation networks and robotic systems are of critical societal importance. These systems are often challenging to control because they tend to have complicated and poorly understood dynamics, sometimes with legacy components are built over a long period of time [21]. Therefore detailed models for these systems may not be available or may be intractable to construct. For instance, since buildings account for 40% of the global energy consumption [22], many approaches have been proposed to operate buildings more efficiently by controlling their heating, ventilation, and air conditioning (HVAC) systems [23]. Most of these methods, however, suffer from two drawbacks. On one hand, a detailed physics model of a building can be used to accurately describe its behavior, but this model can take years to develop. On the other hand, simple control algorithms have been developed by using linear (RC circuit) models [24] to represent buildings, but the performance of these models may be poor since the building dynamics can be far from linear [25].

In this paper, we leverage the availability of data to strike a balance between requiring painstaking manual construction of physics based models and the risk of not capturing rich and complex system dynamics through models that are too simplistic. In recent years—with the growing deployment of sensors in physical and robotics systems—large amount of operational data have been collected, such as in smart buildings [69], legged robotics [70] and manipulators [71]. Using these data, the system dynamics can be learned directly and then automatically updated at periodic intervals. One popular method is to parameterize these complex system dynamics using deep neural networks to capturing complex relationships [37, 72], yet few research investigated how to integrate deep

learning models into real-time closed-loop control of physical systems.

In this paper we tackle the modeling accuracy and control tractability tradeoff by building on the input convex neural networks (ICNN) in [8] to both represent system dynamics and to find optimal control policies. By making the neural network convex from input to output, we are able to obtain *both good predictive accuracies and tractable computational optimization problems.* The overall methodology is shown in Fig. 4.1. Our proposed method (shown in Fig. 4.1 (b)) firstly utilizes an input convex network model to learn the system dynamics and then computes the best control decisions via solving a convex model predictive control (MPC) problem, which is tractable and has optimality guarantees. This is different from existing methods that uses model-free end-to-end controller which directly maps input to output (shown in Fig. 4.1 (a)). Another major contribution of our work is that we explicitly prove that ICNN can represent all convex functions and systems dynamics, and is *exponentially* more efficient than widely used convex piecewise linear approximations [73].

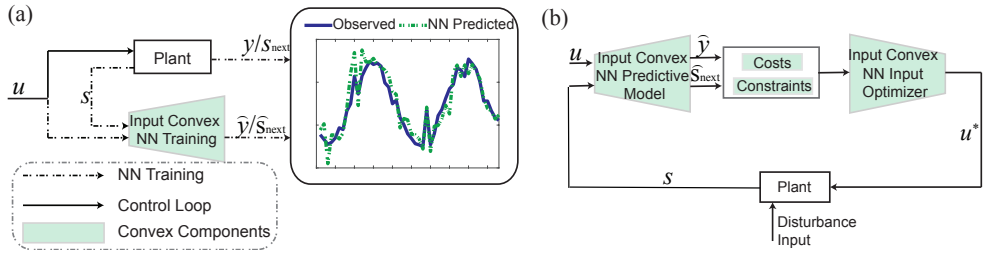

Figure 4.1: Our proposed model-based method, (a) an input convex neural network is first trained to learn the system dynamics, then (b) we solve a convex predictive control problem to find the optimal actions which are input convex neural networks' inputs. The optimization steps are also based on objectives and dynamics constraints represented by the trained networks.

4.2 Connections to Related Work

4.2.1 Model-Based Reinforcement Learning

In many literature, model-based reinforcement learning algorithms share some similarities with optimal control methods. Model-based control algorithms often involve two stages – system identification and controller design. For the system identification stage, the goal is to learn a fixed form of system model to minimize some prediction error [74]. Most efficient model-based control algorithms have used a relatively simple function estimator for the system dynamics identification [75], such as linear model [24] and Gaussian processes [70, 71]. These simplified models are sample-efficient to learn, and can be nicely incorporated in the sub-sequent optimal control problems. However, such simple models may not have enough representation capacity in modeling large-scale or high-dimension systems with nonlinear dynamics. Deep neural networks (DNNs) feature powerful representation capability, while the main challenge of using DNNs for system identification is that such models are typically highly non-linear and non-convex [76], which causes great difficulty for following decision making. A recent work from [75] is close in spirit as our proposed method. Similarly, the authors use a model-based approach for robotics control, where they first fit a neural network for the system dynamics and then use the fitted network in an MPC loop. However, since [75] use conventional NN for system identification, they cannot solve the MPC problem to global optimality. Our work shows how the proposed ICNN control algorithm achieves the benefits from both sides of the world. The optimization with respect to inputs can be implemented using off-the-shelf deep learning optimizers, while we are able to obtain good identification accuracies and tractable computational optimization problems by using proposed method at the same time.

4.2.2 System Identification and Optimal Control

On the other hand, the surge of deep learning methods also shed light on directly using deep neural networks as approximators for controllers under unknown and complex environments. Control and decision-making have used deep learning mainly in model-free end-to-end controller settings (shown

in Fig. 4.1 (a)), such as sequential decision making in game [77], robotics manipulation [78, 79], and control of cyber-physical systems [80, 81]. However, much of the success relies heavily on a reinforcement learning setup where the optimal state-action relationship can be learned via a large number of samples. While in practice there are many engineering challenges. For instance, many physical systems do not fit into the reinforcement learning process, where both the sample collection is limited by real-time operations, and there are physical model constraints hard to represent efficiently. Specifically designed reinforcement learning methods have to take issues and methods such as distribution shift [82], off-policy learning [83] and contrastive learning [84] into account.

Such challenges are also prevailing in many energy systems. For example, in the distribution grid operation tasks, full information of grid topology and line parameters are always needed for model-based controller design. To mitigate such burdens on system identification, in [85], a controller is proposed by using linearized system model estimated from advanced metering data. While in [86], a reinforcement learning based controller is directly applied to learn the voltage regulation policy based on power system measurements. However, the optimality and feasibility of such controller are not discussed, and do not leverage the large body of literature on the convexity of the voltage regulation problem. Indeed, in the research community of machine learning and especially reinforcement learning, “learning to control" remains to be a challenging problem when considering physical model constraints and optimality guarantees [87, 88].

4.3 Convex Neural Network

The following proposition states a simple sufficient condition for a neural network to be input convex:

Proposition 1. *The feedforward neural network in Fig. 4.2(a) is convex from input to output given that all weights between layers* $\mathbf{W}_{1:k}$ *and weights in the “passthrough” layers* $\mathbf{D}_{2:k}$ *are non-negative, and all of the activation functions are convex and nondecreasing (e.g. ReLU).*

This proposition follows directly from composition of convex functions [31]. Although it allows

for any increasing convex activation functions, in this paper we work with the popular ReLU activation function. Two notable additions in ICNN compared with conventional feedforward neural networks are: 1) Addition of the direct *"passthrough" layers* connecting inputs to hidden layers and conventional *feedforward layers* connecting hidden layers for better representation power. 2) the expanded inputs that include both $\mathbf{u}$ and $-\mathbf{u}$. The proposed ICNN structure is shown in Fig. 4.2(a). Note that such construction guarantees that the network is convex and non-decreasing with respect to the expanded inputs $\hat{\mathbf{u}} = \begin{bmatrix} \mathbf{u} \\ -\mathbf{u} \end{bmatrix}$, while the output can achieve either decreasing or non-decreasing functions over $\mathbf{u}$.

A simple example that demonstrates how the proposed ICNN can be used to fit a convex function comes form fitting the $|u|$ function. This function is convex and both decreasing and increasing. Let the activation function be $ReLU(\cdot) = \max(\cdot, 0)$. We can write $|u| = -u + 2ReLU(u)$. However, in this representation, we need a negative weight, the -1 in front of u, and this would be troublesome if we compose several networks together. In our proposed ICNN structure with all positive weights and input negation duplicates, we can write $|u| = v + 2ReLU(u)$, where we impose a constraint $v = -u$. Such doubline on the number of input variables may potentially make the network harder to train. Yet during control, having all of the weights positive maintains the convexity between inputs and outputs even if multiple steps are considered which will be discussed in Section 4.3.1. The constraint $v = -u$ is linear and can be easily included in any convex optimization.

The structure of the input convex neural network (ICNN) structure in Proposition 1 is motivated by the structure in [8] but modified to be more suitable to control of dynamical systems. In [8] it only requires $\mathbf{W}_{2:k}$ to be non-negative while having no restrictions on weights $\mathbf{W}_1$ and $\mathbf{D}_{2:k}$. Our construction achieves the exact representation by expanding the inputs to include both $\mathbf{u}$ ($\in \mathbb{R}^d$) and $-\mathbf{u}$. Then any negative weights in $\mathbf{W}_1$ and $\mathbf{D}_{2:k}$ in [8]'s ICNN structure is set to zero and its negation (which is positive) is added as the weight for corresponding $-\mathbf{u}$. The reason for our construction is to allow the network to be "rolled out in time" when we are dealing with dynamical systems and multiple networks need to be composed together.

An simple example that demonstrates how the proposed ICNN can be used to fit a convex func-

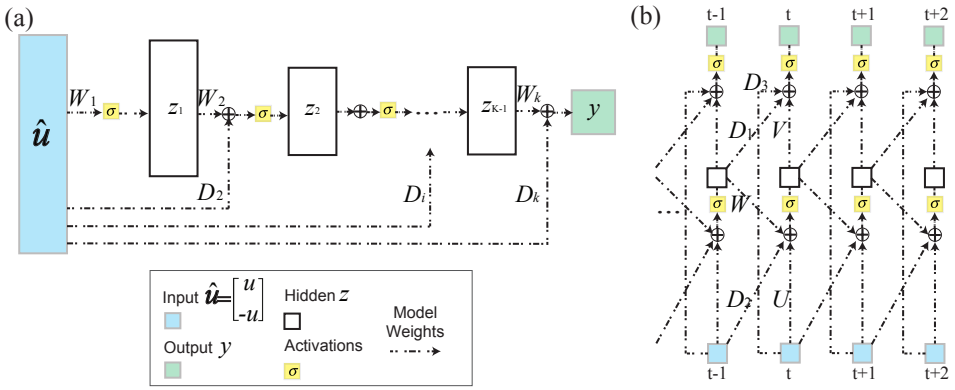

Figure 4.2: Input convex neural network. (a) Input convex feed-forward neural networks (ICNN). One notable addition is the direct "passthrough" layers $\mathbf{D}_{2:k}$ that connect the inputs to hidden units for better model representation ability. (b) The proposed input convex recurrent neural networks (ICRNN) architectures. In our control settings, we keep all weights in both networks nonnegative, while expanding the inputs with $-\mathbf{u}$.

tion comes form fitting the $|u|$ function. This function is convex and both decreasing and increasing. Let the activation function be $ReLU(\cdot) = \max(\cdot, 0)$. We can write $|u| = -u + 2ReLU(u)$ [8]. However, in this representation, we need a negative weight, the -1 in front of u, and this would be troublesome if we compose several networks together. In our proposed ICNN structure with all positive weights and input negation duplicates, we can write $|u| = v + 2ReLU(u)$, where we impose a constraint $v = -u$. Such doubline on the number of input variables may potentially make the network harder to train. Yet during control, having all of the weights positive maintains the convexity between inputs and outputs even if multiple steps are considered which will be discussed in Section 4.3.1. The constraint $v = -u$ is linear and can be easily included in any convex optimization.

This proposition follows directly from composition of convex functions [31]. Although it allows for any increasing convex activation functions, in this paper we work with the popular ReLU activation function. Two notable additions in ICNN compared with conventional feedforward neural networks are: 1) Addition of the direct *"passthrough" layers* connecting inputs to hidden layers and conventional *feedforward layers* connecting hidden layers for better representation power. 2) the expanded inputs that include both $\mathbf{u}$ and $-\mathbf{u}$. The proposed ICNN structure is shown in Fig. 4.2(a).

Note that such construction guarantees that the network is convex and non-decreasing with respect to the expanded inputs $\hat{\mathbf{u}} = \begin{bmatrix} \mathbf{u} \\ -\mathbf{u} \end{bmatrix}$, while the output can achieve either decreasing or non-decreasing functions over $\mathbf{u}$.

Sp far, we have shown ICNN allows us to use neural networks in decision-making processes by guaranteeing the solution is unique and globally optimal. Since many complex input and output relationships can be learned through deep neural networks, it is natural to consider using the learned network in an optimization problem in the form of

$$\min_{\mathbf{u}} f(\mathbf{u}; \mathbf{W}) \tag{4.1a}$$

$$\text{s.t. } \mathbf{u} \in \mathscr{U}, \tag{4.1b}$$

where $\mathscr{U}$ is a convex feasible space. Then if f is an ICNN, optimizing over $\mathbf{u}$ is a convex problem, which can be solved efficiently to global optimality. Note that we will always duplicate the variables by introducing $\mathbf{v} = -\mathbf{u}$, but again this does not change the convexity of the problem. Of course, since the weights of the network are restricted to be nonnegative, the performance of the network (e.g., classification) may be worse. A common thread we observe in this paper is that trading off classification performance with tractability can be preferable.

4.3.1 Closed-Loop Control with Temporal Dynamics

In addition to the single-shot optimization problem in (4.1), we are interested in optimally controlling a dynamical system. To model the temporal dependency of the system dynamics, we propose to use recurrent neural networks (instead of feed-forward neural networks). Recurrent networks carry an internal state of the system, which introduces coupling with previous inputs to the system. Fig. 4.2(b) shows the proposed input convex recurrent neural networks (ICRNN) structure. This network maps from input $\hat{\mathbf{u}}$ to output y with memory unit $\mathbf{z}$ according to the following Eq. (4.2),

$$\mathbf{z}_t = \sigma_1(\mathbf{U}\hat{\mathbf{u}}_t + \mathbf{W}\mathbf{z}_{t-1} + \mathbf{D}_2\hat{\mathbf{u}}_{t-1}), \tag{4.2}$$

$$y_t = \sigma_2(\mathbf{V}\mathbf{z}_t + \mathbf{D}_1\mathbf{z}_{t-1} + \mathbf{D}_3\hat{\mathbf{u}}_t), \tag{4.3}$$

where $\hat{\mathbf{u}} = \begin{bmatrix} \mathbf{u} \\ -\mathbf{u} \end{bmatrix}$, and D_1, D_2, D_3 are added direct "passthrough" layers for augmenting representation power. If we unroll the dynamics with respect to time, we have $y_t = f(\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, ..., \hat{\mathbf{u}}_t; \theta)$ where $\theta = [\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{D_1}, \mathbf{D_2}, \mathbf{D_3}]$ are network parameters, and σ_1, σ_2 denote the nonlinear activation functions. The next proposition states a sufficient condition for the network to be input convex.

Proposition 2. *The network shown in Fig. 4.2(b) is a convex function from inputs to output if all weights U, V, W, D_1, D_2, D_3 are non-negative, and all activation functions are convex and nondecreasing (e.g. ReLU).*

The proof of this proposition again follows directly from the composition rule of convex functions. Similarly to the ICNN case, by expanding the inputs vector to include both $\mathbf{u}$ and $-\mathbf{u}$ and restricting all weights to be non-negative, the resulted ICRNN structure is a convex and non-decreasing mapping from inputs to output.

The proposed ICRNN structure can be leveraged to represent system dynamics for close-loop control. Consider a physical system with discrete-time dynamics, at time step t, let's define $\mathbf{s}_t$ as the *system states*, $\mathbf{u}_t$ as the *control actions*, and y_t as the *system output*. For example, for the real-time control of a building system, $\mathbf{s}_t$ includes the room temperature, humidity, etc; $\mathbf{u}_t$ denotes the building appliance scheduling, room temperature set-points, etc; and output y_t is the building energy consumption. In addition, there maybe exogenous variables that impact the output of the system, for example, outside temperature will impact the energy consumption of the building. However, since the exogenous variables are not impacted by any of the control actions we take, we suppress them in the formulation below. The time evolution of a system is described by

$$y_t = f(\mathbf{s}_t, \mathbf{u}_t), \tag{4.4a}$$

$$\mathbf{s}_{t+1} = g(\mathbf{s}_t, \mathbf{u}_t) \tag{4.4b}$$

where (4.4b) describes the coupling between the current inputs to the future system states. Physical systems described by (4.4) may have significant inertia in the sense that the outcome of any control actions is delayed in time and there are significant couplings across time periods.

4.4 Optimal Control via ICNN

Since we use ICRNNs to represent both the system dynamics $g(\cdot)$ and the output $f(\cdot)$, the control variable $\mathbf{u}$ expands as $\hat{\mathbf{u}}$. The optimal receding horizon control problem at time t can be written as,

$$\underset{\mathbf{u}_t,\mathbf{u}_{t+1},\ldots,\mathbf{u}_{t+T}}{\text{minimize}} \quad C(\hat{\mathbf{x}},\mathbf{y}) = \sum_{\tau=t}^{t+T} J(\hat{\mathbf{x}}_\tau, y_\tau) \tag{4.5a}$$

$$\text{subject to} \quad y_\tau = f(\hat{\mathbf{x}}_{\tau-n_w}, \hat{\mathbf{x}}_{\tau-n_w+1}, \ldots, \hat{\mathbf{x}}_\tau), \forall \tau \in [t, t+T] \tag{4.5b}$$

$$\mathbf{s}_\tau = g(\hat{\mathbf{x}}_{\tau-n_w}, \hat{\mathbf{x}}_{\tau-n_w+1}, \ldots, \hat{\mathbf{x}}_{\tau-1}, \hat{\mathbf{u}}_\tau), \forall \tau \in [t, t+T] \tag{4.5c}$$

$$\hat{\mathbf{x}}_\tau = \begin{bmatrix} \mathbf{s}_\tau \\ \hat{\mathbf{u}}_\tau \end{bmatrix}, \ \hat{\mathbf{u}}_\tau = \begin{bmatrix} \mathbf{u}_\tau \\ \mathbf{v}_\tau \end{bmatrix}, \forall \tau \in [t, t+T] \tag{4.5d}$$

$$\mathbf{v}_\tau = -\mathbf{u}_\tau, \forall \tau \in [t, t+T] \tag{4.5e}$$

$$\mathbf{s}_\tau \in \mathscr{S}_{feasible}, \forall \tau \in [t, t+T] \tag{4.5f}$$

$$\mathbf{u}_\tau \in \mathscr{U}_{feasible}, \forall \tau \in [t, t+T] \tag{4.5g}$$

where a new variable $\hat{\mathbf{x}} = [\mathbf{s}_t, \hat{\mathbf{u}}_t]$ is introduced for notational simplicity, which called *system inputs*. It is the collection of system states $\mathbf{s}_t$ and duplicated control actions $\mathbf{u}_t$ and $-\mathbf{u}_t$, therefore ensuring the mapping from $\mathbf{u}_t$ to any future states and outputs remains convex. $J(\hat{\mathbf{x}}_\tau, y_\tau)$ is the control system cost incurs at time τ, that is a function of both the system inputs $\hat{\mathbf{x}}_\tau$ and output y_τ. The functions $f(\cdot)$ and $g(\cdot)$ in Eq. (4.5b)-(4.5c) are parameterized as ICRNNs, which represent the system dynamics from sequence of inputs $(\hat{\mathbf{x}}_{\tau-n_w}, \hat{\mathbf{x}}_{\tau-n_w+1}, \ldots, \hat{\mathbf{x}}_\tau)$ to the system output y_τ, and the dynamics from control actions to system states, respectively. n_w is the memory window length of the recurrent neural network. The equations (4.5d) and (4.5e) duplicate the input variables $\mathbf{u}$ and enforce the consistency condition between $\mathbf{u}$ and its negation $\mathbf{v}$. Lastly, (4.5f) and (4.5g) are the constraints on feasible system states and control actions respectively. Note that as a general formulation, we do not include the duplication tricks on state variables, so the dynamics fitted by (4.5b) and (4.5c) are non-decreasing over state space, which are not equivalent to those dynamics represented by linear systems. However, since we are not restricting the control space, and we have explicitly included multiple previous states in the system transition dynamics, so the non-decreasing constraint over

state space should not restrict the representation capacity by much. In Section.4.5 we theoretically prove the representability of proposed networks.

Optimization problem in (4.5) is a convex optimization with respect to (w.r.t.) inputs $\mathbf{u} = [\mathbf{u}_t, ..., \mathbf{u}_{t+T}]$, provided the cost function $J(\hat{\mathbf{x}}_\tau, y_\tau) = J(\mathbf{s}_\tau, \hat{\mathbf{u}}_\tau, y_\tau)$ is convex w.r.t. $\hat{\mathbf{u}}_\tau$, and convex, nondecreasing w.r.t. $\mathbf{s}_\tau$ and y_τ. A problem is convex if and only if both the objective function and constraints are convex. In the above problem, $J(\mathbf{s}_\tau, \hat{\mathbf{u}}_\tau, y_\tau)$ is convex and nondecreasing w.r.t. $\mathbf{s}_\tau$ and y_τ; $\mathbf{s}_\tau$ and y_τ are parameterized as ICRNNs, i.e., (4.5a) and (4.5b), such that they are convex w.r.t. $\hat{\mathbf{u}}_\tau$. Therefore following the composition rule of convex functions, the objective function is convex w.r.t. inputs $\mathbf{u} = [\mathbf{u}_t, ..., \mathbf{u}_{t+T}]$. Besides, all the equality constraints (4.5d) and (4.5e) are affine. Suppose both the state feasibile set (4.5f) and action feasibile set (4.5g) are convex, the overall optimization is convex.

The convexity of the problem in (4.5) guarantees that it can be solved efficiently and optimally using gradient descend method. Since both the objective function (4.5a) and the constraints (4.5b)-(4.5c) are parameterized as neural networks, and their gradients can be calculated via back-propagation with the modification where cost is propagated to the input rather than the weights of the network. For implementation, the gradients can be convinently calculated via existing modules such as Tensorflow viaback-propagation. Let $\mathbf{u}^* = \{\mathbf{u}_t^*, \mathbf{u}_{t+1}^*, ..., \mathbf{u}_{t+T}^*\}$ be the optimal solution of the optimization problem at time t. Then the first element of $\mathbf{u}^*$ is implemented to the real-time system control, that is $\mathbf{u}_t^*$. The optimization problem is repeated at time $t+1$, based on the updated state prediction using $\mathbf{u}_t^*$, yielding a model predictive control strategy.

4.5 Representation Power and Efficiency of ICNN

Besides the computational tractability of the input convex networks, as an system identification model, we are also interested its predictive accuracies and capacity. This section provides theoretical analysis on the representation ability and efficiency of input convex neural networks.

4.5.1 Representation power of input convex neural network

Definition 1. *Given a function $f : \mathbb{R}^d \to \mathbb{R}$, we say that the function $\hat{f}$ approximate f within ε if $|f(\mathbf{x}) - \hat{f}(\mathbf{x})| \leq \varepsilon$ for all $\mathbf{x}$ in the domain of f.*

Theorem 1. *[Representation power of ICNN] For any Lipschitz convex function over a compact domain, there exists a neural network with nonnegative weights and ReLU activation functions that approximates it within ε.*

Lemma 1. *Given a continuous Lipschitz convex function $f : \mathbb{R}^d \to \mathbb{R}$ with compact domain and $\varepsilon > 0$, it can be approximated within ε by maximum of a finite number of affine functions. That is, there exists $\hat{f}(\mathbf{x}) = \max_{i=1,\ldots,N}\{\boldsymbol{\mu_i}^T \mathbf{x} + b_i\}$ such that $|f(\mathbf{x}) - \hat{f}(\mathbf{x})| \leq \varepsilon$ for all $\mathbf{x} \in domf$.*

Sketch of proof for Theorem 1. Supposing Lemma 1 is true, the proof of Theorem 1 boils down to showing that neural network with nonnegative weights and ReLU activation functions can exactly represent a maximum of affine functions. The proof is constructive. We first construct a neural network with ReLU activation functions and both positive and negative weights, then we show that the weights between different layers of the network can be restricted to be nonnegative by a simple duplication trick. Specifically, since the weights in the input layer and passthrough layers in the ICNN can be negative, we simply add a negation of each input variable (e.g. both $\mathbf{x}$ and $-\mathbf{x}$ are given as inputs) to the network. These variables need satisfy a consistency constraint since one is the negation of the other. Since this constraint is linear, it preserves the convexity of optimization problems.

This proof is similar in spirit to theorems in [89, 90]. The key new result is a simpler construction than the one used in [89] and the restriction to nonnegative weights between the layers. □

Similar to Theorem 1, an analogous result about the representation power of ICRNN can be shown for systems with convex dynamics. Given a dynamical system described by rolled out system dynamics $y_t = f(\mathbf{x}_1, \ldots, \mathbf{x}_t)$ is convex, then there exists a recurrent neural network with nonnegative weights and ReLU activation functions that approximates it within ε. A broad range of systems can be captured by this model. For example, the linear quadratic (Gaussian) regulator problem can be

described using a ICRNN if we identify y as the cost of the regulator [91, 92].[1] An example of a nonlinear system is the control of electrochemical batteries. It can be shown from first principles that the degradation of these types of batteries is convex in their charge and discharge actions [93] and our framework offers a powerful data-driven way to control batteries found in electric vehicles, cell phones, and power systems.

4.5.2 ICNN vs. convex piecewise linear fitting

In the proof of Theorem 1, we first approximate a convex function by a maximum of affine functions then construct a neural network according to this maximum. Then a natural question is why learn a neural network and not directly the affine functions in the maximum? This approach was taken in [73], where a convex piecewise-linear function (max of affine functions) are directly learned from data through a regression problem.

A key reason that we propose to use ICNN (or ICRNN) to fit a function rather than directly finding a maximum of affine functions is that the former is a much more efficient parameterization than the latter. As stated in Theorem 2, a maximum of K affine functions can be represented by an ICNN with K layers, where each layer only requires a single ReLU activation function. However, given a single layer ICNN with K ReLU activation functions, it may take a maximum of 2^K affine functions to represent it exactly. Therefore in practice, it would be much easier to train a good ICNN than finding a good set of affine functions.

Theorem 2. *[Efficiency of Representation]*

1. *Let $f_{ICNN} : \mathbb{R}^d \to \mathbb{R}$ be an input convex neural network with K ReLU activation functions. Then $\Omega(2^K)$ functions are required to represent f_{ICNN} using a max of affine functions.*
2. *Let $f_{CPL} : \mathbb{R}^d \to \mathbb{R}$ be a max of K affine functions. Then $O(K)$ activation functions are sufficient to represent f_{CPL} exactly with an ICNN.*

Proof. The second statement of Theorem 2 directly follows the construction in the proof of Theorem

[1] It's important to note that y is usually used as the system output of a linear system, but in our context, we are using it to refer to the quadratic cost with respect to the system states and the control input.

1, which shows that a maximum of K affine functions can be represent by a K-layer ICNN (with a single ReLU function in each layer). So it remains to show the first statement of Theorem 2.

To show that a maximum of affine functions can require exponential number of pieces to approximate a function specified by an ICNN with K activation functions, consider a network with 1 hidden layer of K nodes and the weights of direct "passthrough" layers are set to 0:

$$f_{ICNN}(\mathbf{x}) = \sum_{i=1}^{K} w_{1i}\sigma(\mathbf{w}_{0i}^T\mathbf{x} + b_i)\,, \tag{4.6}$$

It contains $3K$ parameters: $\mathbf{w}_{0i}$, w_{1i} and b_i, where $\mathbf{w}_{0i} \in \mathbb{R}^d$ and $w_{1i}, b_i \in \mathbb{R}$.

In order to represent the same function by a maximum of affine functions, we need to assess the value of every activation unit $\sigma(\mathbf{w}_{0i}^T\mathbf{x} + b_i)$. If $\mathbf{w}_{0i}^T\mathbf{x} + b_i \geq 0$, $\sigma(\mathbf{w}_{0i}^T\mathbf{x} + b_i) = \mathbf{w}_{0i}^T\mathbf{x} + b_i$; otherwise, $\sigma(\mathbf{w}_{0i}^T\mathbf{x} + b_i) = 0$. In total, we have 2^K potential combinations of piecewise-linear function, including

$$\begin{aligned}
L_1 &= \left(\sum_{i=1}^{K} w_{1i}\mathbf{w}_{0i}\right)^T \mathbf{x} + \sum_{i=1}^{K} w_{1i}b_i\,, \text{if all } \mathbf{w}_{0i}^T\mathbf{x} + b_i \geq 0 \\
L_2 &= \left(\sum_{i=2}^{K} w_{1i}\mathbf{w}_{0i}\right)^T \mathbf{x} + \sum_{i=2}^{K} w_{1i}b_i\,, \text{if } \mathbf{w}_{01}^T\mathbf{x} + b_1 < 0 \text{ and all other } \mathbf{w}_{0i}^T\mathbf{x} + b_i \geq 0 \\
L_3 &= \left(w_{11}\mathbf{w}_{01} + \sum_{i=3}^{K} w_{1i}\mathbf{w}_{0i}\right)^T \mathbf{x} + w_{1i}b_i + \sum_{i=3}^{K} w_{1i}b_i\,, \text{if } \mathbf{w}_{02}^T\mathbf{x} + b_2 < 0 \text{ and other } \mathbf{w}_{0i}^T\mathbf{x} + b_i \geq 0 \\
&\cdots\cdots\,, \\
L_{2^K} &= 0\,, \text{ if all } \mathbf{w}_{0i}^T\mathbf{x} + b_i < 0.
\end{aligned}$$

So the following maximum over 2^K pieces is required to represent the single linear ICNN:

$$\max\{L_1, L_2, ..., L_{2^K}\}.$$

□

4.6 Experiments

In this section, we demonstrate the effectiveness of ICNN and ICRNN by presenting experimental results on two decision-making problems: energy management of large-scale commercial buildings

and the distribution network voltage control, respectively. The proposed method can be used as a flexible building block in decision making problems, where we use ICRNN to model the relationship between temperature setpoints and building energy consumption, while we use ICNN to represent the relationship between the active and reactive power injections to nodal voltage magnitude deviation. Both examples demonstrate that proposed method: 1) closes the loop between system modeling and controller design; 2) is lightweight and sample-efficient; 3) achieves more generalizable and better control performances compared with previous model-based reinforcement learning and simplified linear systems.

4.6.1 Building Energy Management

We consider the real-time control problem of building's HVAC (heating, ventilation, and air conditioning) system to reduce its energy consumption. Building energy management remains to be a hard problem in control area. The exact system dynamics are unknown and hard to model due to the complex heating transfer dynamics, time-varying environments and the scale of the system in terms of states and actions [94]. At time t, we assume the building's running profile $\mathbf{x}_t := [\mathbf{s}_t, \mathbf{u}_t]$ is available, where $\mathbf{s}_t$ denotes building system states, including outside temperature, room temperature measurements, zone occupancies and etc. $\mathbf{u}_t$ denotes a collection of control actions such as room temperature set points and appliance schedule. Output is the electricity consumption P_t.

This is a model predictive control problem in the sense that we want to find the best control inputs that minimize the overall energy consumption of building by looking ahead several time steps. To achieve this goal, we firstly learn an ICRNN model $f(\cdot)$ of the building dynamics, which is trained to minimize the error between P_t and $f(\mathbf{x}_{t-n_w}, ..., \mathbf{x}_t)$, while n_w denotes the memory window

of recurrent neural networks. Then we solve:

$$\underset{\mathbf{u}_t,\ldots,\mathbf{u}_{t+T}}{\text{minimize}} \quad \sum_{\tau=t}^{t+T} f(\mathbf{x}_{\tau-n_w},\ldots,\mathbf{x}_\tau) \tag{4.7a}$$

$$\text{subject to} \quad \mathbf{s}_\tau = g(\mathbf{x}_{\tau-n_w},\ldots,\mathbf{x}_{\tau-1},\mathbf{u}_\tau), \forall \tau \in [t,t+T] \tag{4.7b}$$

$$\underline{\mathbf{u}}_\tau \leq \mathbf{u}_\tau \leq \overline{\mathbf{u}}_\tau, \forall \tau \in [t,t+T] \tag{4.7c}$$

$$\underline{\mathbf{s}}_\tau \leq \mathbf{s}_\tau \leq \bar{\mathbf{s}}_\tau, \forall \tau \in [t,t+T] \tag{4.7d}$$

where the objective (4.7a) is minimizing the total energy consumption in future T steps (T is the model predictive control horizon), and (4.7b) is used for modeling building states, in which $g(\cdot)$ are parameterized as ICRNNs. Note that the formulation (4.7) is also flexible with different loss functions. For instance, in practice, we could reuse trained dynamics model (4.7b), and integrate electricity prices into the overall objective so that we could directly learn real-time actions to minimize electricity bills (please refer to Appendix E for more results). The constraints on control actions $\mathbf{u}_t$ and system states $\mathbf{s}_t$ are given in (4.7c) and (4.7d). For instance, the temperature set points as well as real measurements should not exceed user-defined comfort regions.

To test the performance of the proposed method, we set up a 12-story large office building, which is a reference EnergyPlus commercial building model from US Department of Energy (DoE) [2], with a total floor area of $498,584$ square feet which is divided into 16 separate zones. By using the whole year's weather profile, we simulate the building running through the year and record ($\mathbf{x}_t$, P_t) with a resolution of 10 minutes. We use 10 months' data to train the ICRNN and subsequent 2 months' data for testing. We use 39 building system state variables $\mathbf{s}_t$ (uncontrollable), along with 16 control variables $\mathbf{u}_t$. Output is a single value of building energy consumption at each time step. We set the model predictive control horizon $T = 36$ (six hours). We employ an ICRNN with recurrent layer of dimension 200 to fit the building input-output dynamics $f(\cdot)$. The model is trained to minimize the MSE between its predictions and the actual building energy consumption using stochastic gradient descent. We use the same network structure and training scheme to fit state transition dynamics $g(\cdot)$.

[2]Energyplus is an open-source whole-building energy modeling software, which is developed by US DoE for standard building energy simulation

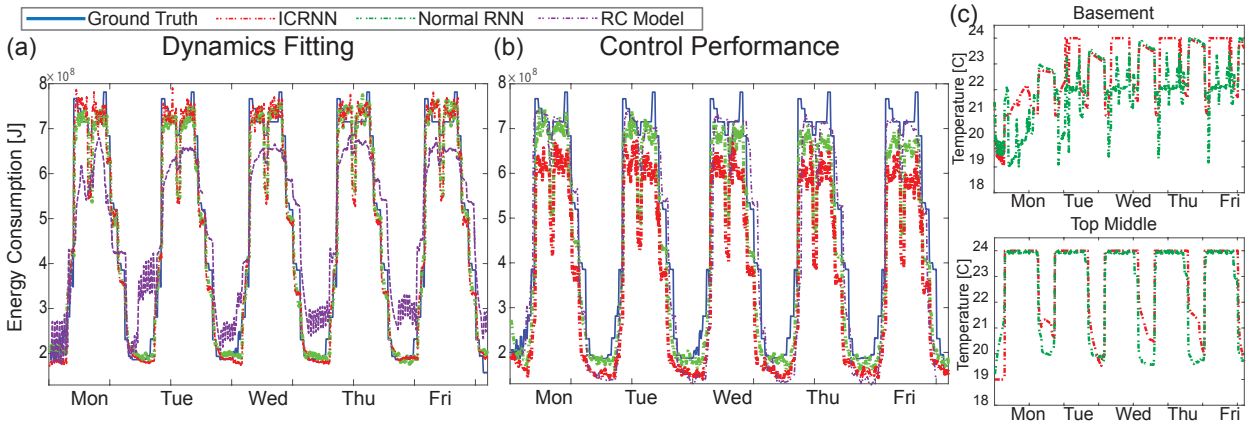

Figure 4.3: Results for constrained optimization of building energy management. (a) ICRNN is able to model the building dynamics as accurately as conventional RNN; (b) Compared to conventional RNN model, ICRNN finds control actions which lead to 11.52% more of energy savings, and (c) ICRNN provides stable control actions while decisions generated by conventional RNN vary dramatically.

Baseline We set the model-based forecasting and optimization benchmark using an linear resistor-circuit (RC) circuit model to represent the heat transfer in building systems, and solve for the optimal control actions via MPC [24]. At each step, MPC algorithm takes into account the forecasted states of the building based on the fitted RC model and implements the current step control actions. We also compare the performance of ICRNN against the conventionally trained RNN in terms of building dynamics fitting performance and control performance. To solve the MPC problem with conventional RNN models, we also use gradient-based method with respect to controls. However, since conventional RNN models are generally not convex from input to output, there is no guarantee to reach a global optimum (or even a local one).

Results In terms of the fitting performance, ICRNN provides a competitive result compared to conventional RNN model. The overall test root mean square error (RMSE) is 0.054 for ICRNN and 0.051 for conventional RNN, both of which are much smaller than the error made by RC model (0.240). Fig. 4.3(a) shows the fitting performance on 5 working days in test data. This illustrates the good performance of ICRNN in modeling building HVAC system dynamics. Then by using the learned ICRNN model of building dynamics, we obtain the suggested room control

actions u_t^* by solving the optimal building control problem (4.7). As shown in Fig. 4.3(b), with the same constraints on building temperature interval of $[19°C, 24°C]$, the building energy consumption is reduced by 23.25% after implementing the new temperature set points calculated by ICRNN. On the contrary, since there is no guarantee for finding optimal control actions by optimizing over conventional RNN's input, the control solutions given by conventional RNN could only reduce 11.73% of electricity. Solutions given by RC model only saves 4.07% of electricity. More importantly, in Fig. 4.3(c) we demonstrate the control actions outputted by our method against MPC with conventional RNN in two randomly selected building zones, the building basement and top floor central area. It shows that our proposed approach is able to find a group of stable control actions for the building system control. While in the conventional RNN case, it generates control set points which have undesirable, drastic variations.

4.6.2 Distribution Grid Voltage Regulation

Motivation and Problem Formulation

Voltage regulation in distribution networks has played an important role to maintain acceptable voltage magnitudes at all buses. The higher penetration of distributed energy resources (DERs), for example rooftop PV and electric vehicles, could lead to fast voltage fluctuations in distribution networks [95]. To complement slow time-scale control of discrete devices such as tap-changing transformers and switched capacitors, reactive power injections via the inverter-based distributed resources are often proposed for fast time-scale voltage regulations [96].

Recall the voltage regulation problem stated in Chapter 2, the relaxed line flow constraints together with the voltage regulation objective formulates a Second Order Cone Program:

Remark 3. *The relaxed voltage control problem with regulation objective* (2.7a)*, reactive power injection constraint* (2.7b)*, power flow constraints* (2.4a)-(2.4c) *and* (2.8) *is convex. Under many circumstances, this relaxation is tight, see [45, 44].*

In order to further simplify the analysis of the original voltage control problem (2.7), many linearized power flow models are adopted, while Simplified Distflow model is widely used [97, 42],

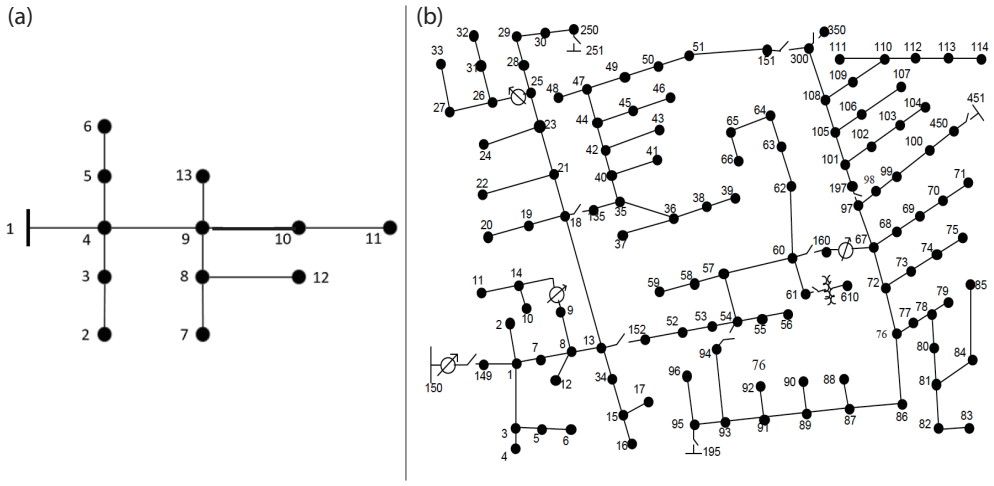

Figure 4.4: Schematic diagram of (a). IEEE 13-bus test feeder and (b). IEEE 123-bus test feeder. Reference buses are 1 and 149 respectively.

which sets l_{ik} to be zeros, and approximates $V_i^2 - V_k^2$ by $2(V_i - V_k)$:

$$-p_k = p_{ik} - \sum_{l:(k,l)\in\mathscr{E}} p_{kl} \tag{4.8a}$$

$$-q_k = q_{ik} - \sum_{l:(k,l)\in\mathscr{E}} q_{kl} \tag{4.8b}$$

$$V_i - V_k = r_{ik}p_{ik} + x_{ik}q_{ik}. \tag{4.8c}$$

Similarly, by replacing (2.7c) with the linearized version (4.8), we are also able to solve the voltage regulation as a convex optimization problem.

However, considering the increasing variability of load and generation in distribution networks, voltage regulation based on linearized approximation model (4.8) may not be accurate enough to represent the true distribution network models, while the resulting control signals of reactive power injections may not be optimal when applied in the real distribution networks.

In summary, to solve voltage regulation as a convex optimization, all these aforementioned approaches require the exact information on line parameters (e.g., line impedances) and network topology. Unfortunately, due to the lack of observability of distribution systems, directly learning the topology is hard without PMU data [98, 99].

Given the practical challenges of voltage regulation, we want to design an optimal controller that satisfies following requirements:

- The controller must learn an accurate representation of the power injections to nodal voltage magnitudes;

- Such representation is easy to be integrated into the optimization framework.

Intuitively, we are trying to design and find functions $\left|V_i - V_{i,0}\right| = f_i(\mathbf{p},\mathbf{q}), i = 1,...,N$, which could accurately represent the relationship from active and reactive power injections to nodal voltage magnitude deviation. By leveraging historical smart meter data to fit f_i, we want to see if the fitted model could represent the underlying grid. More importantly, if f_i is a convex function from $\mathbf{p},\mathbf{q}$ to $\left|V_i - V_{i,0}\right|$, then the following problem

$$\min_{\mathbf{q}} \quad \sum_{i=1}^{N} \alpha_i |V_i - V_{i,0}| \tag{4.9a}$$

$$\text{s.t.} \quad \underline{\mathbf{q}} \leq \mathbf{q} \leq \overline{\mathbf{q}} \tag{4.9b}$$

$$\left|V_i - V_{i,0}\right| = f_i(\mathbf{p},\mathbf{q}) \tag{4.9c}$$

is still a tractable convex optimization problem. Note that we integrate voltage magnitude deviations constraint (4.9c) into the voltage regulation framework, which is a general formulation to make sure once f_i is convex, (4.9) is a convex optimization problem. Such formulation is comparable to previous formulations by either treating voltage magnitude deviations as the optimization objective [44] or as box constraints [100, 43].

Because of the convexity results [45], restricting f_i to be convex leads to optimal solutions. Following the similar spirits in the building energy management task, we parameterize (4.9c) using ICNN $h_\theta(\mathbf{p},\mathbf{q})$, with each dimension of the neural network's output correponds to the voltage magnitude deviation for bus i.

Simulation Results

For both 13-bus and 123-bus system, we use AC power flow model (2.4) to generate $10,000$ instances of simulation data composed of $\{\mathbf{p}, \mathbf{q}, \mathbf{V}\}$. We assume both the distribution network topology and line parameters are not revealed to the optimization algorithm, except when the optimal SOCP is used as a baseline. We use one year's load data from the University of Washington Seattle campus for training and test dataset generation. We allow plus/minus 20% of reactive power injections at each node as control inputs. We develop three algorithms and compare their performances for the IEEE 13-bus and 123-bus test feeders as shown in Fig. 4.4:

- *Linear Model:* We consider using a linear model to fit the unknown dynamics from active and reactive power to the deviations between nodal voltage and the nominal voltage. Such linearized models have been widely used in power systems literature [42, 43];

- *Neural Networks Model:* We construct standard three-layer and four-layer neural networks for the 13-bus and 123-bus cases, respectively. We tune the parameters of neural networks (e.g., number of neurons, learning rate) and stop the training process once the fitting performance on validation data converges;

- *Input Convex Neural Networks:* We keep the number of layers and matrices $W_i, i = 1, ..., k$ the same dimension as those of neural networks models, but add direct layers $D_i, i = 2, ..., k$ correspondingly. We constrain network weights $W_{2:k}$ to be non-negative during training.

To fit the parameters of neural network models, we use mean squared error as the loss function during training. To solve the voltage regulation problem (2.7), we set $\alpha_i, i = 1, ..., k$ in (4.9) to be 1 in our simulation cases. When α is not equal to 1, we could adapt the optimization problem using weighted sum of voltage deviation correspondingly. Note that we could also flexibly use alternative loss terms or add reactive power costs to the objective function (2.7a), as long as they are convex functions over reactive power injections. All the implementations are conducted on a MacBook Pro with 2.4GHz Intel Quad Core i5.

Simulation Network	IEEE 13-Bus			
Model	SOCP	Linear	NN	ICNN
Model Fitting MAE	-	9.93%	3.45%	3.86%
Regulated voltage out of 3% tolerance	3.46%	8.65%	7.88%	4.71%
Regulated voltage out of 5% tolerance	0.47%	7.89%	6.86%	1.05%
Computation Time (per instance/s)	0.9684	0.2022	0.3137	0.2512
Simulation Network	IEEE 123-Bus			
Model	SOCP	Linear	NN	ICNN
Model Fitting MAE	-	12.98%	3.56%	4.25%
Regulated voltage out of 3% tolerance	3.61%	21.46%	14.04%	7.51%
Regulated voltage out of 5% tolerance	0.72%	19.19%	9.65%	1.64%
Computation Time (per instance/s)	4.041	0.2712	0.6297	0.4302

Table 4.1: Comparison between SOCP, Linear model, Neural Networks model, and Input Convex Neural Networks model for IEEE 13-bus and IEEE 123-bus systems.

To benchmark the performance of the proposed algorithms under unknown topology and parameters, we also follow [45] to relax $l_{ij} \geq \frac{P_{ij}^2+Q_{ij}^2}{V_i^2}$ in the Dist-flow equations, and use the same validation datasets to solve the resulting convex SOCP. We calculate the optimal reactive power injections along with the resulting voltage profiles. We use CVX to solve the SOCP and linearized models [101], and use Tensorflow to set up and optimize over NN and ICNN models [38].

Algorithm 3 Distributed Voltage Regulation via ICNN

Require: Input dataset $\{\mathbf{p}, \mathbf{q}\}$

Require: Number of buses N

Require: Trained ICNN model $h_\theta(\mathbf{p}, \mathbf{q})$

while $\left| q_i^{(i)} - q_i^{(j)} \right| > \delta$ for any $(i, j) \in \mathcal{E}$ **do**

for $i = 1, ..., N$ **do**

Solve subproblem (4.16) for i based on λ_{ij}

Get value q_i^i and $q_j^i, \forall j \sim i$

end for

Update λ_{ij} based on q_i^i and q_i^j using (4.15)

end while

Nodal optimal reactive power injection q_i

Estimation Accuracy

We firstly validate that ICNN can be used as a proxy for power flow equations, and predict the nodal voltage magnitude deviations. By using $8,000$ training instances, the ICNN can predict the voltage deviations on the validation instances accurately. As shown in Table 4.1, the mean absolute error (MAE) of ICNN fitting are smaller than 4.3% in both test systems, which are comparable to 3.45% and 3.56% by using neural networks. This is also illustrated in Fig. 4.5, where under different load levels throughout 24 hours, the ICNN can predict all the nodal voltages accurately. More importantly, linear model's fitting performances are over 2 times worse than the neural networks counterparts. We later show such performances on model tractability and fitting errors would also impact the controller performances.

Voltage Regulation Performance

In Figure 4.5, we show the regulated voltage using ICNN in the IEEE 13-bus case. Under this day's load profile, we are able to regulate node 4's voltage magnitude within $\pm 4\%$ per unit with

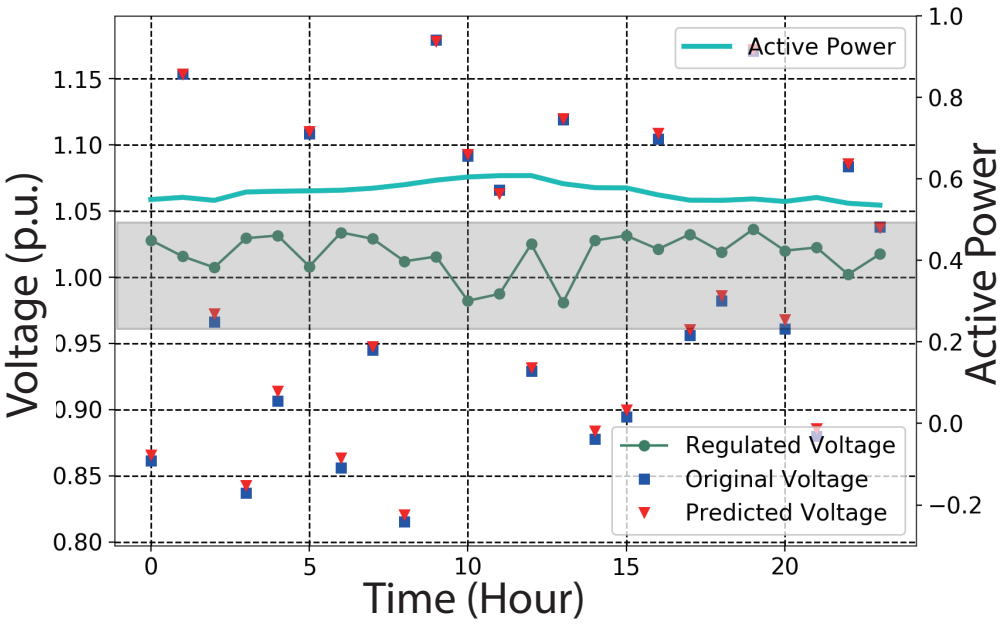

Figure 4.5: Example of voltage regulation over a daily variation for the 13-bus test feeder. The voltage of bus 4 is shown. With ICNN accurately predicting voltages (red triangle), it could regulate voltage within 4% of nominal values (grey box) under varying load level throughout the day.

constrained reactive power injections (Equation 4.9b). In Figure 4.6, we show that the mean and variance on each bus's voltage deviations using three models for the 13-bus feeder. On the one hand, with similar fitting performances, ICNN outperforms the standard neural network in regulating nodal voltages. This is due to the fact that neural networks may have many local minima, and the NN-based controller can not find the optimal reactive power injections. On the other hand, even though linear model provides a easier venue for solving optimization problem, it suffers from inaccurate modeling of the underlying distribution grids, and the regulated bus voltages have greater level of fluctuations. Similar observations also hold in the 123-bus test case, where in Fig. 4.7 we show the nodal voltage comparison using three models, and voltage regulated by ICNN are constrained to be in a much narrower range. More results on voltage regulation performances are summarized in Table 4.1. Under varying load and power generation profiles, ICNN is able to maintain over 98.3% of nodal voltages within 5% deviations from nominal voltages, which are comparable to SOCP solutions. On the contrary, linear fitted models can not scale to larger system,

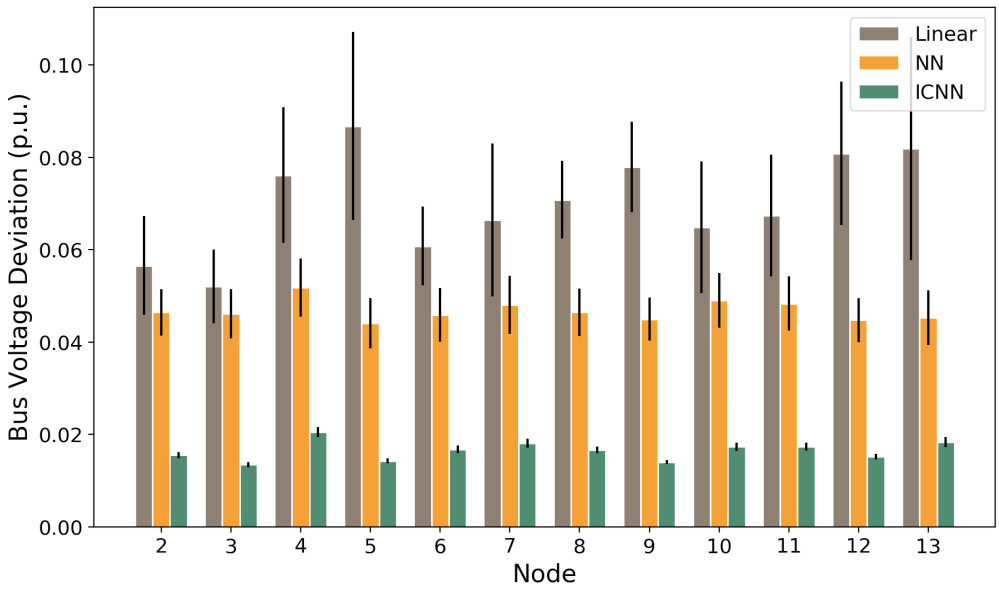

Figure 4.6: Comparisons on nodal voltage deviation bar plots of linear-fitted model, neural network model and input convex neuural network model on IEEE 13-bus system. On average, the mean voltage deviation for ICNN is 4.3 times smaller than linear model, and 2.7 times smaller than standard NN model.

and nearly 20% of voltages are out of 5% tolerance in the 123-bus case.

We also give an analysis on the computation time for each algorithm as shown in last row of Table 4.1. Reported results are averaged over 2,000 testing instances. Compared to linear model, optimization based on ICNN generally takes longer time due to the model complexity, but it is still able to find the optimal solutions within the acceptable time range. Note that we are solving the ICNN optimization problem using our own solver, while solving SOCP and linear model using the off-the-shelf CVX solvers. The optimization solution process involving ICNN can be further accelerated with GPU support in the future. More importantly, in the 13-bus case, ICNN-based optimization is faster than SOCP solver, and it scales to 123-bus case with moderate computation time increases compared to SOCP solver. This makes ICNN as a practical modeling and optimization tool for unknown distribution grids. An interesting observation is that it takes longer for NN to find solutions compared to ICNN, partly due to the fact that gradient-based optimizer is stuck in some local minima in normal NN.

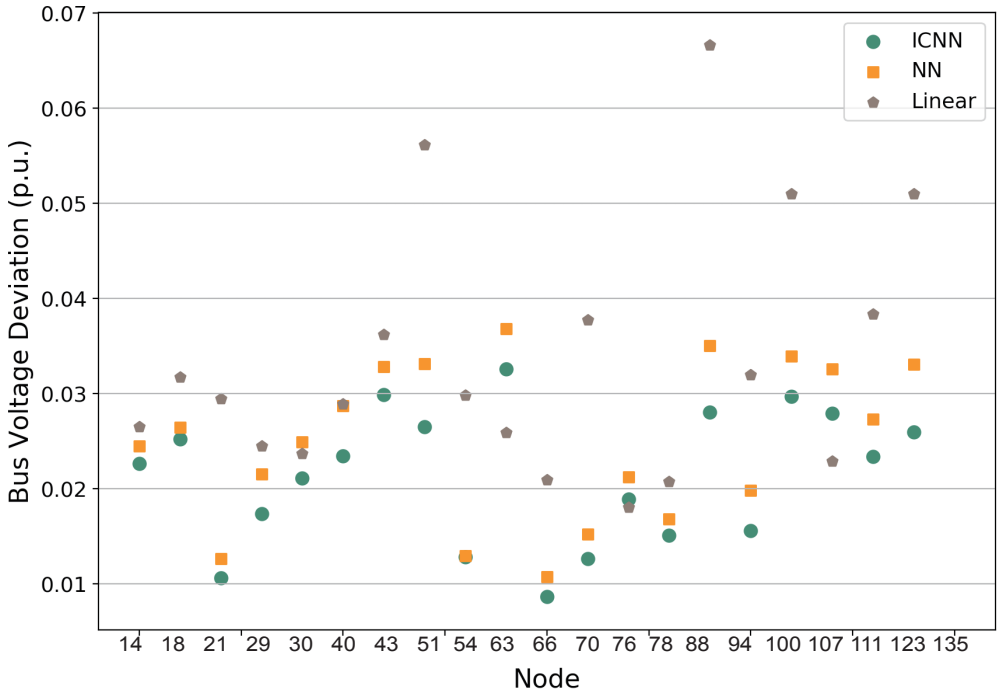

Figure 4.7: Comparisons on 20 randomly selected buses' nodal voltage deviation plots of linear-fitted model, neural network model and input convex neuural network model on IEEE 123-bus system.

Distributed Algorithm

Again, due to the convexity of ICNN, our problem is essentially following the standard formulation of distributed convex optimization problem [102]. This allows us to borrow from a large body of literature on distributed algorithms for convex problems. We are interested in minimizing the sum of their individual convex (potentially non-smooth) objective functions of N interconnected agents:

$$f^* := \min_{q \in Q} \quad \sum_{i=1}^{N} f_i(\mathbf{q}, \mathbf{p}), \tag{4.10}$$

where Q is the feasible set defined by the bounds on reactive power injection. Each function f_i is assumed to be pre-trained and known by agent i beforehand. The goal is to solve the minimization problem in a decentralized fashion, where the agents cooperatively find the optimal reactive power

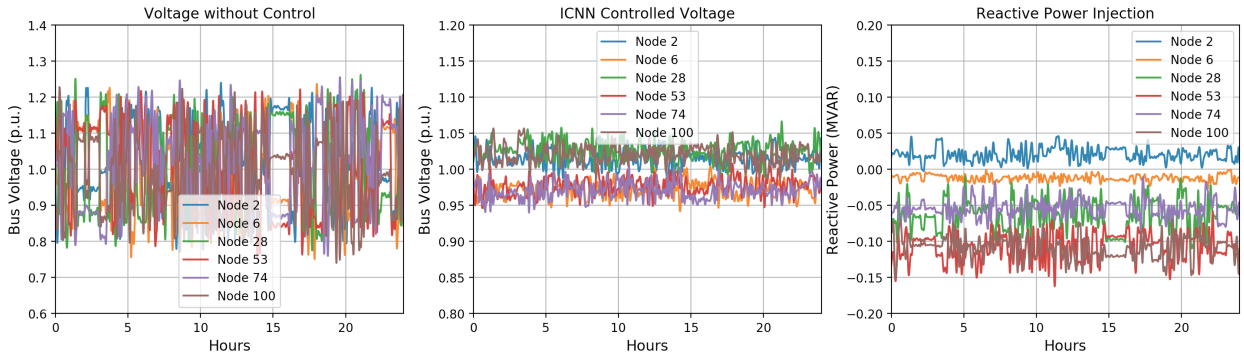

Figure 4.8: Simulation results of proposed distributed algorithm with 24-hour time-varying nodal loads and generations. The left and middle plots show the control performances on nodal voltages, while right plot shows the scale of control actions (reactive power injections).

injection without a central coordinator. The coupling can be written out explicitly as

$$\min_{q^1,\ldots,q^N \in Q} \sum_{i=1}^{N} f_i(\mathbf{q},\mathbf{p}) \tag{4.11a}$$

$$\text{s.t.} \quad q^1 = q^2 = \cdots = q^N \tag{4.11b}$$

Problem (4.11) can be solved via dual decomposition [103, 104]. Let's denote q_j^i as node i's estimate of node j's reactive power injection. Then the augmented Lagrangian L is

$$L\left(q^1,\ldots,q^N,\lambda\right) := \sum_{i=1}^{N} f_i\left(q_1^i,\ldots,q_N^i\right) + \sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_{ij}\left(q_i^i - q_i^j\right) \tag{4.12}$$

with corresponding dual function is the infinmum over primal variables:

$$q(\lambda) := \inf_{q^1,\ldots,q^N \in Q} L\left(q^1,\ldots,q^N,\lambda\right) \tag{4.13}$$

Note that $\lambda_{ii} = 0$, and we can also rewrite $\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_{ij}(q_i^i - q_i^j) = \sum_{(i,j)\in\mathcal{E}} \left[\lambda_{ij}\left(q_i^{(i)} - q_i^{(j)}\right) + \lambda_{ji}\left(q_j^{(i)} - q_j^{(j)}\right)\right]$, so there is only $\lambda_{ij} \neq 0$ when there is a communication between neighboring nodes. Then for each node, there is a decomposable subproblem

$$\phi^i(\lambda) := \inf_{q^i \in Q} f_i\left(q_1^i, q_2^i, \ldots, q_N^i\right) + \sum_{j=1}^{N} \lambda_{ij} q_i^i - \sum_{j=1}^{N} \lambda_{ji} q_j^i \tag{4.14}$$

Via gradient ascent, at iteration $t+1$, the update rule for dual variable is

$$\lambda_{ij}(t+1) = \lambda_{ij}(t) + \alpha_t \left(q_i^i(t) - q_i^j(t) \right) \tag{4.15}$$

which essentially says that by exchanging Lagrangian multipliers for neighboring nodes, there shall reach consensus on the dual variables. When all λ_{ij} are optimal, q_i^i will be equal to q_j^i for all (i, j) pair, and the duality gap is zero.

So at each iteration, at each agent i, by using trained ICNN to parameterize f_i, it solves the subproblem simultaneously:

$$\min_{q_i} \quad f_i\left(q_1^i, q_2^i, \ldots, q_N^i\right) + \sum_{j=1}^{N} \lambda_{ij} q_i^i - \sum_{j=1}^{N} \lambda_{ji} q_j^i \tag{4.16a}$$

$$\text{s.t.} \quad \underline{q} \leq q_i \leq \overline{q} \tag{4.16b}$$

Since the optimization problem is convex, iterating this process (Algorithm 4.6.2) guarantees convergence to the global optimal solution.

In Figure 4.8 we show the performance of proposed learning and control algorithm under the distributed settings, where the underlying topology is unknown to nodes' controllers, while local communication is allowed for neighboring controllers in the communication graph. We plot the voltage and reactive power injection profiles on 6 randomly selected nodes under 24-hour's varying active power injections. For each node, the voltage magnitude deviation can be controlled effectively using ICNN based controller. As we assume control components at all buses for the distribution grid can supply or consume at most 0.2 MVar reactive power, it is shown in the right plot of Figure 4.8 that such constraints are satisfied at all buses at all times.

4.7 Conclusion

In this chapter, we have shown a pipeline for connecting learning system dynamics and designing control actions via an input convex neural network. The ICNN can be served as a flexible unit for general optimal control tasks. Apart from the architecture design for neural networks, we have theoretically shown that ICNN has the promise to approximate any convex functions, while it is

much more efficient than the normal practice of the model. We have empirically demonstrated that ICNN can play a significant role in building energy management and voltage regulation tasks.

The idea of enforcing convexity can not only help design control systems, it can also be used for many tasks with both theoretical and performance guarantees. In the next chapter, we show how ICNN can serve as a modeling unit for solving optimization problems. It is also interesting to explore how such convex models can help decision-making in uncertain or fast-changing environments.

Chapter 5

LEARNING TO MAKE DECISIONS FOR ENERGY DISPATCH

5.1 Motivation

Despite decades of studies, LPs can still face computational challenges in some settings. Often these challenges are due to the increased stochasticity and the low-latency requirements of real-time applications. The motivating application of this chapter is in power systems, where intermittent and random renewable resources are being integrated into the grid at all levels [105, 106]. These new resources can create new flow patterns that fundamentally transform how systems operate. Inadequate planning for sudden events can lead to significant losses of welfare, as demonstrated by the recent rolling blackouts in Texas [107] and California [108].

The DCOPF problem has been studied extensively in the last sixty years and is a workhorse of the power industry [109]. If the generator cost is linear, the DCOPF is a linear program. If the costs are quadratic, then it is a quadratic program with linear constraints. Both types of optimization problems can be solved efficiently by a variety of algorithms, which have been implemented in a number of software packages [110, 111]. Today, a DCOPF problem can be solved quickly for fairly large networks [112, 113].

Because of the uncertainties brought by the renewables on many of the nodes, the number of generation and load scenarios that need to be considered are starting to grow exponentially [114, 115, 116]. Even if each scenario under consideration takes less than a second to solve using modern solvers, not all of them can be completed within the required time period. For example, if one instance of DCOPF can be solved in 0.5 seconds, then solving it for 2,000 scenarios would take more than 15 minutes, while outside the 5 minutes time resolution that real-time DCOPF are performed in practice. Therefore, using neural networks (NNs) to learn the mapping between input load profiles and the corresponding optimal generation outputs has gained significant attention, since making

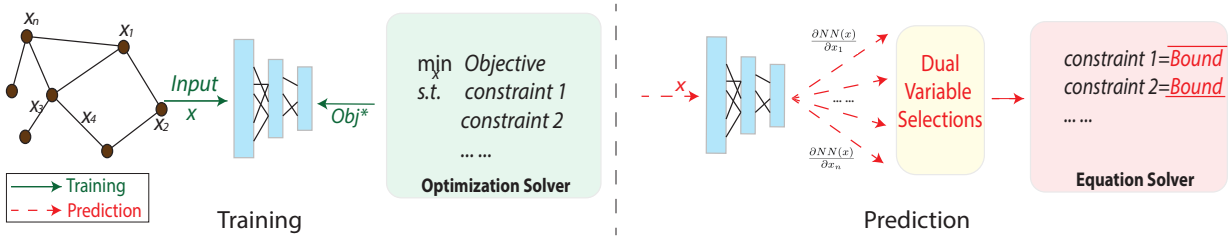

Figure 5.1: The schematic of our proposed Neural Decoder for solving OPF problems. A neural network is trained to predict the optimal objective value; during implementation for solving OPF, the network's gradient is interpreted as a noisy codeword for active constraints, and a linear equation is solved to obtain optimal solutions.

inference via a trained architecture can be potentially orders of magnitude faster than an iterative solver [117, 118].

Even if each scenario under consideration takes 0.1 seconds to solve using modern solvers, not all of them can be completed within the required time period. Therefore, using neural networks to learn the mapping between input load profiles and the corresponding optimal generation outputs has gained significant attention, since making inference via a trained architecture can be potentially orders of magnitude faster than an iterative solver [117, 118].

The learning-theoretic question becomes whether neural networks (NNs) can be used to directly make decisions in optimization problems, and in particular, *whether it can learn the solutions of LPs as a function of the changes in the problem data*. Surprisingly, the answer to this question has been largely negative. In [119, 120], convex optimization problems are embedded as layers to standard neural networks, with the rationale that directly learning the mapping from problem data to solutions is difficult, especially when learned solutions need to satisfy all of the constraints.

In this chapter, we answer the question in the affirmative for LPs and a family of quadratic programming problems. Instead of viewing it as an end-to-end learning task, we leverage the rich algorithmic understanding of LPs as well as the economic interpretation of the primal and dual variables to offer a novel solution architecture. Our workflow is shown in Figure 5.1. Concretely, we construct a neural network that takes the net load at each node as the input and outputs the optimal system cost (a scalar). Of course, the optimal value of the cost is not the solution of the

LP. Rather, using the neural network, we compute the *gradient* of the cost with respect to the net loads. Identifying these as the *dual variables*, we use them to predict the binding nodal and line constraints. Once these constraints are identified, the optimal solution is given by solving a simple linear system of equations. The overall procedure can be seen as an efficient surrogate learning model for optimization solvers.

5.2 Connections to Related Work

Solving LP in Engineering Applications Generic LP solvers have been optimized to the point that they provide the state-of-the-art algorithms for many applications. For example, power system operators use CPLEX or Gurobi for their dispatch problems [121, 122, 123, 124]. Specialized problems such as max flow have dedicated algorithms that have faster theoretical guarantees [125], but general network flow problems will often use generic solvers [33, 126]. A key challenge is that many applications now require problem to be solved repeatedly with stringent latency constraints [127].

Learning to Solve Optimization Our work falls under the category of using machine learning to solve optimization problems, which has a long history dating back to at least [128, 129]. A line of research considers using deep learning to solve combinatorial optimization problems typically by developing new heuristics [130, 131]. This chapter is closer to the work that tries to solve convex optimization problem by directly using supervised learning to find the optimal mapping from input data to the optimal solution, with applications in power systems [118, 132], scheduling [133, 134], and resource management [135]. These algorithms can be regarded as end-to-end behavior cloning of expert policies [136].

The challenge of using end-to-end models is that they can suffer from compounding errors and poor generalization performances, especially when hard constraints need to be satisfied. In [119], a small LP problem with 3 variables is shown to be hard to learn via supervised learning. Our work is also related to [137, 138], where authors proposed to predict the set of active constraints at optimality. The learned neural networks in these settings have difficulties generalizing because of the high dimensionality of their outputs and need training data to essentially cover input space.

In [106, 139, 140], the authors describe reinforcement learning formulation to model the interaction between learners' decisions and rewards, but there is no guarantee the solutions satisfy all constraints. In [141, 120, 142, 143], differentiable convex optimization layers provide convenient venues of designing end-to-end models, yet they still use standard solvers at runtime and do not achieve our goal of speeding up computations in time/resource-constrained applications.

5.3 Solving Optimization using a Neural Network

We now first look into the DCOPF problem with linear costs. Under such assumptions, we can view the DCOPF in (2.6) as a linear programming problem. At the optimal solution of an LP, a constraint is called active if it is an equality constraint or an inequality constraint that binds. In this section, we assume the LP is not degenerative, and the key result (Theorem 2) we use from linear programming is that the number of active constraints is exactly the same as the number of decision variables in the problem.

Theorem 4. *For a nondegenerative linear programming problem with p optimization variables, the optimal solution is determined by p linearly independent active constraints.*

This theorem states that if we know which constraints are active at the optimal, we can simply solve linear equations to find the optimal solution satisfying all constraints instead of solving an optimization problem. This is especially advantageous for network flow problems, where the number of constraints are typically much larger than the number of optimization variables. The proof of the theorem is standard and can be found in many textbooks on linear programming [33].

While solving the optimization problem is to find the mapping from the load vectors $\boldsymbol{\ell}$ to the optimal value of the objective function. We denote this function as $J^*(\boldsymbol{\ell})$, which implicitly encodes the feasibility and optimality conditions. Since the output of J is a scalar and is continuous in $\boldsymbol{\ell}$, it is much easier to learn than the vector of optimal solutions or the active constraints. Of course, we are rarely satisfied about knowing only the optimal cost. But this function has a rich structure that can be exploited using the following theorem:

Theorem 5. *Let $\boldsymbol{\mu}^* \in \mathbb{R}^n$ denote the optimal dual variables associated with equality constraints* (2.6d)*. If for a given $\boldsymbol{\ell}$* (2.6) *is feasible, then $\nabla_{\boldsymbol{\ell}} J^* = \boldsymbol{\mu}^*$. Furthermore, the optimal solution for* (2.6) *is associated with the following active/inactive constraints:*

$$x_i = \begin{cases} \bar{x}_i, & \text{if } \mu_i^* - c_i > 0 \\ 0 & \text{if } \mu_i^* - c_i < 0 \\ (0, \bar{x}_i), & \text{otherwise} \end{cases} \tag{5.1}$$

5.3.1 Proof of Theorem 5

Proof. The proof follows from standard duality theory. We require feasibility of the input $\boldsymbol{\ell}$ to rule out the dual being unbounded. The Lagrangian of (2.6) is given by

$$\begin{aligned} L(\boldsymbol{\ell}, \mathbf{x}, \mathbf{f}, \boldsymbol{\mu}, \bar{\boldsymbol{\lambda}}, \underline{\boldsymbol{\lambda}}, \bar{\boldsymbol{\nu}}, \underline{\boldsymbol{\nu}}) = \mathbf{c}^T \mathbf{x} + \boldsymbol{\mu}^T (\boldsymbol{\ell} - \mathbf{x} - \tilde{\mathbf{A}}\mathbf{f}) + \\ \bar{\boldsymbol{\lambda}}^T (\mathbf{K}\mathbf{f} - \bar{\mathbf{f}}) - \underline{\boldsymbol{\lambda}}^T (\mathbf{K}\mathbf{f} + \underline{\mathbf{f}}) - \underline{\boldsymbol{\nu}}^T \mathbf{x} + \bar{\boldsymbol{\nu}}^T (\mathbf{x} - \bar{\mathbf{x}}) \end{aligned} \tag{5.2}$$

where $\bar{\boldsymbol{\nu}}$ and $\underline{\boldsymbol{\nu}}$ are the dual variables associated with the capacity constraints in (2.6b), $\bar{\boldsymbol{\lambda}}$ and $\underline{\boldsymbol{\lambda}}$ are the dual variables associated flow capacities in (2.6c), and $\boldsymbol{\mu}$ is the dual variable associated with the equality constraint (2.6d). The dual variables $\bar{\boldsymbol{\nu}}$, $\underline{\boldsymbol{\nu}}$, $\bar{\boldsymbol{\lambda}}$ and $\underline{\boldsymbol{\lambda}}$ are nonnegative.

If $\mathbf{x}^*, \mathbf{f}^*, \boldsymbol{\mu}^*, \bar{\boldsymbol{\lambda}}^*, \underline{\boldsymbol{\lambda}}^*, \bar{\boldsymbol{\nu}}^*, \underline{\boldsymbol{\nu}}^*$ are the optimal primal and dual solutions, then by strong duality

$$J^*(\boldsymbol{\ell}) = L(\boldsymbol{\ell}, \mathbf{x}^*, \mathbf{f}^*, \boldsymbol{\mu}^*, \bar{\boldsymbol{\lambda}}^*, \underline{\boldsymbol{\lambda}}^*, \bar{\boldsymbol{\nu}}^*, \underline{\boldsymbol{\nu}}^*)$$

and differentiating (5.2) gives $\nabla_{\boldsymbol{\ell}} J^* = \boldsymbol{\mu}^*$.

We now analyze the relationships between binding inequality constraints and optimal dual variables associated with equality constraints. With known $\boldsymbol{\mu}^*$ and separable constraints (5.4b)(5.4c), the dual problem is decomposable to the optimization problem consisting $\bar{\boldsymbol{\lambda}}, \underline{\boldsymbol{\lambda}}$ and $\bar{\boldsymbol{\nu}}, \underline{\boldsymbol{\nu}}$ respectively. The optimization problem involving $\bar{\boldsymbol{\nu}}, \underline{\boldsymbol{\nu}}$ can be reformulated as

$$\min_{\bar{\boldsymbol{\nu}}, \underline{\boldsymbol{\nu}}} \quad \bar{\boldsymbol{\nu}}^T \bar{\mathbf{x}} \tag{5.3a}$$

$$\text{s.t.} \quad \bar{\boldsymbol{\nu}} - \underline{\boldsymbol{\nu}} = \boldsymbol{\mu}^* - \mathbf{c}, \tag{5.3b}$$

$$\bar{\boldsymbol{\nu}} \geq \mathbf{0}, \underline{\boldsymbol{\nu}} \geq \mathbf{0} \tag{5.3c}$$

which can read out the result without solving explicitly. Given that $\bar{\boldsymbol{\nu}}, \underline{\boldsymbol{\nu}} \geq \mathbf{0}$, for node i, if $\mu_i^* - c_i > 0$, $\bar{\nu}_i > 0, \underline{\nu}_i = 0$, and $x_i = \bar{x}_i$; if $\mu_i^* - c_i < 0$, $\bar{\nu}_i = 0, \underline{\nu}_i > 0$, and $x_i = 0$; if $\mu_i^* - c_i = 0$, $\bar{\nu}_i = \underline{\nu}_i = 0$ and corresponding x_i is not binding. □

There are also economic interpretations of the dual variable $\boldsymbol{\mu}$. If the LMP at a bus ishigher than the cost of the bus, then the upper bound must bebinding. Conversely, if the LMP is lower than the cost at a bus, the lower bound must be binding. Otherwise, the generator is on the margin and neither bounds are binding.

Theorem 5 states if $\boldsymbol{\mu}^*$ is known, then all active constraints at each of the node can be readily "decoded" by exploiting the parameters of original optimization problem. Of course, two questions remain: 1) how do we find the active *flow constraints* and 2) how do we find $\boldsymbol{\mu}^*$?

5.3.2 Finding Active Flow Constraints

We now describe how to find the active constraints of the edge flows in (2.6c) which can be generalized to cases in the family of network flow problems (2.5c). Let $\bar{\boldsymbol{\nu}}$ and $\underline{\boldsymbol{\nu}}$ be the dual variables associated with the capacity constraints in (2.6b), $\bar{\boldsymbol{\lambda}}$ and $\underline{\boldsymbol{\lambda}}$ are the dual variables associated flow capacities in (2.6c), and $\boldsymbol{\mu}$ is the dual variable associated with the equality constraint (2.6d). The dual of (2.6) is:

$$\max_{\boldsymbol{\mu}, \bar{\boldsymbol{\lambda}}, \underline{\boldsymbol{\lambda}}, \bar{\boldsymbol{\nu}}, \underline{\boldsymbol{\nu}}} \quad \boldsymbol{\mu}^T \boldsymbol{\ell} - \underline{\boldsymbol{\lambda}}^T \underline{\mathbf{f}} - \bar{\boldsymbol{\lambda}}^T \bar{\mathbf{f}} - \bar{\boldsymbol{\nu}}^T \bar{\mathbf{x}} \tag{5.4a}$$

$$\text{s.t.} \quad \mathbf{c} - \boldsymbol{\mu} - \underline{\boldsymbol{\nu}} + \bar{\boldsymbol{\nu}} = \mathbf{0} \tag{5.4b}$$

$$-\tilde{\mathbf{A}}^T \boldsymbol{\mu} - \mathbf{K}^T \underline{\boldsymbol{\lambda}} + \mathbf{K}^T \bar{\boldsymbol{\lambda}} = \mathbf{0} \tag{5.4c}$$

$$\bar{\boldsymbol{\nu}} \geq \mathbf{0}, \underline{\boldsymbol{\nu}} \geq \mathbf{0}, \bar{\boldsymbol{\lambda}} \geq \mathbf{0}, \underline{\boldsymbol{\lambda}} \geq \mathbf{0} \tag{5.4d}$$

If $\boldsymbol{\mu}^*$ is given (we will talk about how to learn this in Section 5.5), the dual variables associated with the nodal constraints and the edge flow constraints decouple, where the latter is:

$$\max_{\bar{\boldsymbol{\lambda}},\underline{\boldsymbol{\lambda}}} \quad -\underline{\boldsymbol{\lambda}}^T\underline{\mathbf{f}} - \bar{\boldsymbol{\lambda}}^T\bar{\mathbf{f}} \tag{5.5a}$$

$$\text{s.t.} \quad \mathbf{K}^T(\bar{\boldsymbol{\lambda}} - \underline{\boldsymbol{\lambda}}) = \tilde{\mathbf{A}}^T\boldsymbol{\mu}^* \tag{5.5b}$$

$$\bar{\boldsymbol{\lambda}} \geq \mathbf{0}, \underline{\boldsymbol{\lambda}} \geq \mathbf{0} \tag{5.5c}$$

We cannot directly find line constraint binding conditions due to the coupling between $\bar{\boldsymbol{\lambda}}, \underline{\boldsymbol{\lambda}}$ and $\boldsymbol{\mu}^*$. The following simple lemma is a useful way to rewrite (5.5).

Lemma 2. *Assume that $\underline{\boldsymbol{f}} = \bar{\boldsymbol{f}}$ (symmetric edge capacities). Then the problem* (5.5) *is equivalent to*

$$\min_{\boldsymbol{\upsilon}} \quad ||\boldsymbol{\upsilon}||_1 \tag{5.6a}$$

$$\textit{s.t.} \quad \hat{\mathbf{K}}^T(\boldsymbol{\upsilon}) = \tilde{\mathbf{A}}^T\boldsymbol{\mu}^* \tag{5.6b}$$

where $\upsilon_i = \bar{f}_i \cdot \bar{\lambda}_i - \bar{f}_i \cdot \underline{\lambda_i}$, $|\upsilon_i| = |\bar{f}_i \cdot \bar{\lambda}_i - \bar{f}_i \cdot \underline{\lambda_i}|$, and $\hat{\mathbf{K}} = diag(1/\bar{f}_1, \ldots, 1/\bar{f}_m)\mathbf{K}$.

Proof. Let $\boldsymbol{y} = \bar{\mathbf{f}} \odot \bar{\boldsymbol{\lambda}} + \bar{\mathbf{f}} \odot \underline{\boldsymbol{\lambda}}$ and $\mathbf{v} = \bar{\boldsymbol{\lambda}} - \underline{\boldsymbol{\lambda}}$, where $\odot$ is componentwise multiplication. Then the optimization problem in (5.5) becomes

$$\begin{aligned} \min_{\boldsymbol{y},\mathbf{v}} \quad & \sum_{i=1}^{n} y_i \\ \text{s.t.} \quad & \mathbf{K}^T\mathbf{v} = \tilde{\mathbf{A}}^T\boldsymbol{\mu}^* \\ & \boldsymbol{y} \geq \bar{\mathbf{f}} \odot \mathbf{v} \\ & \boldsymbol{y} \geq -\bar{\mathbf{f}} \odot \mathbf{v}, \end{aligned}$$

where the last two inequalities come from the nonnegativity constraint of $\underline{\boldsymbol{\lambda}}$. Suppose $\boldsymbol{y}, \mathbf{v}$ are optimal solutions. Because we are minimizing the sum of the components of $\boldsymbol{y}$, $y_i = \bar{f}_i \max(v_i, -v_i)$. This is equivalent to $y_i = \bar{f}_i|v_i|$. □

The assumption that the edge capacity is symmetric holds for most undirected networks seen in practice.

For the standard network flow problem in (2.5), the matrix $\mathbf{K}$ is the identity and (5.6b) is a full rank linear equation. Therefore, the optimal $\boldsymbol{\upsilon}$ (and hence the multipliers) can be found by

inspection. However, for the OPF problem in (2.6), the cycle constraints make solving (5.6) less trivial. Fortunately, transformed $\mathcal{L}_1$ minimization problem in (5.6) is extremely well studied, since it falls exactly into the regime of sparse signal recovery in compressed sensing, where signal $\boldsymbol{\upsilon}$ needs to be recovered via observation $\tilde{\mathbf{A}}^T\boldsymbol{\mu}^*$. Note that in sparse recovery, $\mathcal{L}_1$ minimization is a surrogate problem where the ultimate goal is to find the sparsest solution to the problem. For us, (5.6) is the exact problem we want to solve to finish the active constraints identification.

Because there are limited number of active constraints which is equivalent to limited number of nonzero entries in $\boldsymbol{\upsilon}$, while $\hat{\mathbf{K}}$ is very sparse, there are many algorithms that can solve (5.6) extremely efficiently. For example, a family of greedy algorithms such as iterative hard thresholding (IHT) can be used to find $\boldsymbol{\upsilon}$ [144] in a few (fixed) number of iterations.

In our decoding scheme, KKT conditions and dual problems are first utilized to decode a set of dual variables as the status of inequality constraints, while the remaining constraints can be identified by a smaller, subsequent $\mathcal{L}_1$ minimization step.

5.4 Extension to OPF with Quadratic Costs

Following the similar methodology described in Section 5.3, we assume that the LMPs have been learned and describe how it can be used to determine the active constraints when cost is quadratic. If q_i's are not zero, the dual of (2.6) is

$$\max_{\boldsymbol{\mu},\bar{\boldsymbol{\lambda}},\underline{\boldsymbol{\lambda}},\bar{\boldsymbol{\nu}},\underline{\boldsymbol{\nu}}} \quad \boldsymbol{\mu}^T\,\boldsymbol{\ell} - \underline{\boldsymbol{\lambda}}^T\bar{\mathbf{f}} - \bar{\boldsymbol{\lambda}}^T\bar{\mathbf{f}} - \bar{\boldsymbol{\nu}}^T\bar{\mathbf{x}} \tag{5.8a}$$

$$\text{s.t.} \quad \mathbf{Q}\mathbf{x} + \mathbf{c} - \boldsymbol{\mu} - \underline{\boldsymbol{\nu}} + \bar{\boldsymbol{\nu}} = \mathbf{0} \tag{5.8b}$$

$$-\tilde{\mathbf{A}}^T\boldsymbol{\mu} - \mathbf{K}^T\underline{\boldsymbol{\lambda}} + \mathbf{K}^T\bar{\boldsymbol{\lambda}} = \mathbf{0} \tag{5.8c}$$

$$\bar{\boldsymbol{\nu}} \geq \mathbf{0}, \underline{\boldsymbol{\nu}} \geq \mathbf{0}, \bar{\boldsymbol{\lambda}} \geq \mathbf{0}, \underline{\boldsymbol{\lambda}} \geq \mathbf{0}, \tag{5.8d}$$

where $\mathbf{Q}$ is a diagonal matrix with the value of q_i on the i'th diagonal. There are two differences between (5.4) and (5.8). The first is that the constraint associated with the generations, (5.8b), also include the primal variables $\mathbf{x}$. The second is that unlike a linear program, a quadratic program may not have the same number of binding constraints as the variables.

5.4.1 Binding Constraints

Again we assume $\boldsymbol{\mu}$ is known (coming from the gradient of the learned J). We use the following lemma to determine whether a generator constraint is binding:

Lemma 3. *Given the optimal LMP $\boldsymbol{\mu}^*$, x_i is associated with the following active/inactive constraints:*

$$x_i = \begin{cases} \bar{x}_i, & \text{if } \mu_i^* - c_i - 2q_i\bar{x}_i > 0 \\ 0 & \text{if } \mu_i^* - c_i < 0 \\ (0, \bar{x}_i), & \text{otherwise} \end{cases} \tag{5.9}$$

This lemma shows that the binding generation constraints can again be found through simple comparisons.

Proof. The rule in (5.9) follows from simple economic principles. If a generator is marginal, then its LMP is $\mu_i^* = c_i + 2q_i x_i$. Otherwise, a generator is binding at its upper bound if $\mu_i^* > c_i + 2q_i\bar{x}_i$ and at its lower bound if $\mu_i^* < c_i$. □

The binding line constraints can be recovered through the same process as the linear cost case. Once $\boldsymbol{\mu}$ is known, the dual problem associated with $\underline{\boldsymbol{\lambda}}$ and $\overline{\boldsymbol{\lambda}}$ is the same as (5.5).

5.4.2 Finding the Optimal Solutions

Once we identify all of the binding constraints, we can encode it into a matrix of the form $\mathbf{My} = \mathbf{a}$, where $\mathbf{y}$ is the concatenation of $\mathbf{x}$ and $\mathbf{f}$. For a quadratic program, the number of constraints (rows of $\mathbf{M}$) can be less then the number of variables. Therefore, we still need to solve the following optimization problem:

$$\min \frac{1}{2}\mathbf{y}^T\hat{\mathbf{Q}}\mathbf{y} + \hat{\mathbf{c}}^T\mathbf{y} \tag{5.10a}$$

$$\text{s.t. } \mathbf{My} = \mathbf{a}, \tag{5.10b}$$

where $\hat{\mathbf{Q}}$ is a diagonal matrix with $(q_1, \ldots, q_n, 0, \ldots 0)$ on its diagonal and $\hat{\mathbf{c}} = (c_1, \ldots, c_n, 0, \ldots, 0)$. They come from the fact that costs are not assigned to the flows. Fortunately (5.10) can be solved as a linear system:

Lemma 4. *Let $\boldsymbol{\tau}^*$ be the optimal Lagrangian multiplier of* (5.10b)*, then the optimal solution of* (5.10) *is given by the following linear system*

$$\begin{bmatrix} \hat{\mathbf{Q}} & \mathbf{M}^T \\ \mathbf{M} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y}^* \\ \boldsymbol{\tau}^* \end{bmatrix} = \begin{bmatrix} -\hat{\mathbf{c}} \\ \mathbf{a} \end{bmatrix}. \tag{5.11}$$

Proof. The Lagrangian of (5.11) is

$$\frac{1}{2}\mathbf{y}^T\hat{\mathbf{Q}}\mathbf{y} + \hat{\mathbf{c}}^T\mathbf{y} + \boldsymbol{\tau}^T(\mathbf{M}\mathbf{y} - \mathbf{a}),$$

and differentiating with respect to $\mathbf{y}$ gives the stationarity condition

$$\hat{\mathbf{Q}} + \hat{\mathbf{c}} - \mathbf{M}^T\boldsymbol{\tau} = 0. \tag{5.12}$$

Combining (5.12) with the equality constraint (5.10b) gives the system of equations in (5.11). □

The significance of the lemma is that solving the problem with quadratic costs is no more difficult than solving it with linear costs. Once the active constraints are identified, a linear system of equations can be solved to find the optimal solutions.

5.5 The `Neural Decoder` Algorithm

By following Theorem 5, we now describe the learning procedure of J^* along with optimal dual variable $\boldsymbol{\mu}^*$. The training is implemented offline using supervised data by collecting solution data of LP with different $\boldsymbol{\ell}$. We use the optimal cost, optimal multiplier (readily available from most solvers) and the active constraints set as the labeled data.

We do not directly use a neural network to learn $\boldsymbol{\mu}^*$. Since $\boldsymbol{\mu}^*$ is not continuous in $\boldsymbol{\ell}$, learning $\boldsymbol{\mu}^*$ directly becomes a large classification problem, which is difficult because of the large number of possible values. Rather, we fit a neural network (parameterized by θ) $g_\theta(\boldsymbol{\ell})$ that maps optimization model input $\boldsymbol{\ell}$ to the optimal cost. This function is continuous and piecewise linear, with distinct "breakpoints", where the derivative changes value. Therefore, it is naturally parameterized by using ReLU activation units. Let $h(g_\theta(\boldsymbol{\ell}))$ denote the binary vector indicating the active constraints determined by learner's solution based on the procedure described in the last section. The structure

and parameters of the underlying optimization problem provide us with a number of terms in the design of learning objectives:

1. Regression loss defined between $g_\theta(\boldsymbol{\ell})$ and $J^*(\boldsymbol{\ell})$ over the neural network's output;
2. Regression loss for optimal dual variable defined between $\nabla_{\boldsymbol{\ell}} g_\theta(\boldsymbol{\ell})$ and $\boldsymbol{\mu}^*$;
3. Distance loss defined between $h(g_\theta(\boldsymbol{\ell}))$ and $s^*(\boldsymbol{\ell})$, where $s^*(\cdot)$ is the binary vector indicating active constraints at the optimal solution.

The training loss is the sum of these terms

$$\mathcal{L}(\theta) = \|g_\theta(\boldsymbol{\ell}) - J^*(\boldsymbol{\ell})\|_2^2 + \gamma_1 \|\nabla_{\boldsymbol{\ell}} g_\theta(\boldsymbol{\ell}) - \boldsymbol{\mu}^*\|_2^2 + \gamma_2 \|h(g_\theta(\boldsymbol{\ell})) - s^*(\boldsymbol{\ell})\|_H \tag{5.13}$$

where $\|h(g_\theta(\boldsymbol{\ell})) - s^*(\boldsymbol{\ell})\|_H$ is the Hamming distance between active constraint sets[1] [145], and γ_1, γ_2 are penalty parameters.

5.6 Robustness and Generalization Properties of `Neural Decoder`

5.6.1 Robustness to Learning Errors

Because we are dealing with continuous values, the prediction $J(\boldsymbol{\ell})$ of the neural network will invariably have errors. Therefore, it is important that the errors made do not add up and cause incorrect identification of the active constraints. Using an analogy from communication theory, we think of the derivatives of the learned neural network $g_{\boldsymbol{\theta}}(\boldsymbol{\ell})$ as *noisy versions of a codeword*. We are essentially providing an error-correcting approach to decode active constraints that is robust to errors made by the neural network.

Observe that for a given $\boldsymbol{\mu}^*$ and some noise $\boldsymbol{\delta}$, though the optimal solution for the optimization problem (5.5) may be different for $\boldsymbol{\mu}^*$ and $\boldsymbol{\mu}^* + \boldsymbol{\delta}$, the set of active constraints at optimal solutions can remain the same. Therefore, there exists a region around a ground truth optimal dual solution where as long as the noise does not push the solution outside of this region, the set of active constraints remains the same. It turns out these regions are polytopes and easily characterized by solving another linear program. This falls under the well studied area of linear programming sensitivity analysis [33, 146], and is formalized by the next Theorem:

[1] For binary vector $\mathbf{v}$, Hamming norm $\|\mathbf{v}\|_H$ is defined as the number of non-zero entries of vector $\mathbf{v}$

Theorem 6. *Consider a given $\boldsymbol{\ell}$ and its associated optimal dual variables $\boldsymbol{\mu}^*$. There exists a polytope $\mathcal{P}_{\boldsymbol{\mu}^*}$ around $\boldsymbol{\mu}^*$ such that if $\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu}^* \in \mathcal{P}_{\boldsymbol{\mu}^*}$, then the active constraints determined using $\tilde{\boldsymbol{\mu}}$ is correct. The set $\mathcal{P}_{\boldsymbol{\mu}^*}$ is computed by a linear program.*

Proof. We use the following lemma:

Lemma 5. *For an LP problem $\{\mathbf{x}^* = \arg\min_{\mathbf{x}} \mathbf{c}^T\mathbf{x} | \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ with $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\mathbf{x} \in \mathbb{R}^m, m < n$, let $(1), (2), ..., (m)$ be the indices of selected columns of $\mathbf{A}$, such that $\boldsymbol{B}\boldsymbol{x}^* = \mathbf{b}$ with $\mathbf{B} = [\mathbf{a}_{(1)} ... \mathbf{a}_{(m)}] \in \mathbb{R}^{(m\times m)}$ as an invertible basis. For any $\tilde{\mathbf{b}} = \mathbf{b} + \boldsymbol{\delta}$, the optimal solution is still given by $\mathbf{B}$ if and only if $\boldsymbol{B}^{-1}\mathbf{b} + \boldsymbol{B}^{-1}\boldsymbol{\delta} \geq \mathbf{0}$.*

The proof of Theorem 6 follows from Lemma 5 by converting our LP of interest to the standard form. Despite the latter being a known result in linear programming, we have not been able to find the proof of the vector form in existing literature (the component by component result can be found in [33, 146]). Therefore we provide the following proof for completeness.

Proof. To find the region where $\tilde{\mathbf{b}} = \mathbf{b} + \boldsymbol{\delta}$ has the same set of active constraints at the optimal solution, denote the new optimal solution as $\tilde{\mathbf{x}}^*$ that satisfies $\mathbf{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$. So the new optimization problem involving $\tilde{\mathbf{b}}$ becomes

$$\tilde{\mathbf{x}} = \arg\min \quad \mathbf{c}^T\mathbf{x} \tag{5.14a}$$

$$s.t. \quad \mathbf{A}\mathbf{x} = \mathbf{b} + \boldsymbol{\delta} \tag{5.14b}$$

$$\mathbf{x} \geq \mathbf{0} \tag{5.14c}$$

Since we have $\mathbf{x}^* = \mathbf{B}^{-1}\mathbf{b} \geq 0$ with input $\mathbf{b}$, and since $\mathbf{x}^*$ and $\tilde{\mathbf{x}}$ can be represented by the same basis $\mathbf{B} \in \mathbb{R}^{m\times m}$, we have

$$\mathbf{B}^{-1}(\mathbf{b} + \boldsymbol{\delta}) \geq \mathbf{0} \tag{5.15}$$

to ensure the optimal solution's feasibility. So the resulting $\boldsymbol{\delta}$ must satisfy (5.15).

On the other hand, if $\mathbf{B}^{-1}(\mathbf{b} + \boldsymbol{\delta}) \geq 0$, while $\tilde{\mathbf{x}} = \mathbf{B}^{-1}(\mathbf{b} + \boldsymbol{\delta})$ satisfies both equality and inequality constraints. By checking the KKT conditions, it is also the optimal solution, which completes the proof. □

This finishes the proof for the theorem. □

Theorem 6 also provides a fast alternative for identifying $\bar{\boldsymbol{\lambda}}, \underline{\boldsymbol{\lambda}}$ and associated active constraints in (5.5) [117]. We now discuss how to make use of topology and known parameters of the optimization problem to accelerate the robust decoding process of active constraints identification. During training or data generation process, we can keep an offline, finite-length dictionary of unique $\mathcal{M} = \{\tilde{\mathbf{A}}^T \boldsymbol{\mu}^*_{(k)}\}$ along with the set of optimal dual variables $\{[\bar{\boldsymbol{\lambda}}^*_{(k)}, \underline{\boldsymbol{\lambda}}^*_{(k)}]\}$, $\mathcal{P}_{\tilde{\mathbf{A}}^T \boldsymbol{\mu}^*_{(k)}}$ and active constraints, respectively. When $\boldsymbol{\ell}$ comes along with neural network prediction $g_\theta(\boldsymbol{\ell})$ at testing time, with $\boldsymbol{\delta}_{(k)} = \tilde{\mathbf{A}} \nabla_{\boldsymbol{\ell}} g_\theta(\boldsymbol{\ell}) - \tilde{\mathbf{A}}^T \boldsymbol{\mu}^*_{(k)}$, one lookup operation is required to determine if $\boldsymbol{\delta}_{(k)} \in \mathcal{P}_{\tilde{\mathbf{A}}^T \boldsymbol{\mu}^*_{(k)}}$, and we can directly identify the value of $\boldsymbol{\lambda}$. If the dictionary does not include the region of active constraint sets, $\mathcal{L}_1$ minimization problem (5.6) is solved via IHT algorithm. Such design of active constraints prediction contains an error-correcting step by design, where a relatively inaccurate $\nabla_{\boldsymbol{\ell}} g_\theta(\boldsymbol{\ell})$ (compared to $\boldsymbol{\mu}^*$) can still lead to the correct set of binding constraints.

5.6.2 Generalization to Testing Samples

In this section we consider the generalization performance of our proposed method. By generalization, we mean the algorithm should perform well on test samples that were not seen during the training process. We adopt the standard method of analysis here: we assume that the neural network can be trained to zero error on the training samples, then we study the errors for testing samples [147, 148, 149]. We use the following definition as a shorthand:

Definition 2. *We say a neural network is well-trained if it achieves zero loss on the training data.*

Understanding the generalization properties is important because zero training error does not imply small test error. Consider the example given in Fig. 5.2. Suppose we are fitting a piece-wise linear function but only given two points that lie on a line and the training loss is the regression loss. There are infinitely many functions that pass through these points, implying that they have zero training error. Obviously, many of them can have large testing errors for other points on the line. This example also shows that it is not sufficient to just impose convexity or provide gradient information on their own. There are also infinitely many convex functions passing through the

labeled points, and there are infinitely many functions with the right gradients at the given points. To constrain the class of functions to be learned, both convexity and the gradient information are needed.

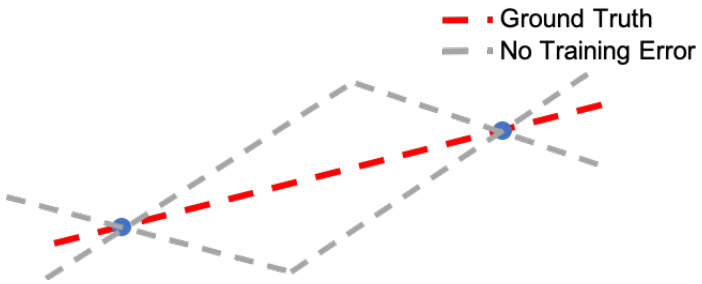

Figure 5.2: Given two points on a line, there are infinitely many piecewise linear functions passing through them. Therefore, small training error (passing through the points) does not necessary imply small generalization error (recovering the line).

Since $J^\star$ is piece-wise linear, we study generalization for two settings. The first is we assume that there are multiple training data within a region where $J^\star$ is linear, and we are interested in the testing performance of a new input from the same region. This setting is about whether the model is constrained enough to not overfit during training. We provide a positive (and simple) answer in Theorem 7.

The second setting is to assume that the test data lies in a region that was unseen during the training process. This question is normally not asked since there is no expectation that learning would be useful for these type of unseen data. However, since there are large number of possible LP regions for DCOPF, it is likely that not all regions would be included in the training data. Therefore, learning algorithms must provide some guarantees on unseen regions. This is especially important as operating conditions change and the historical data are no longer reflective of future scenarios. In Theorem 8, we provide a positive answer showing that the gradients of the unseen region are still bounded.

5.6.3 Generalization for A Linear Region

In this part, we study the case where all training samples have the same value for $\boldsymbol{\mu}$. That is, they come from the same LP region. Let us denote the set of training samples as $\mathscr{D}_{trn}$, and convhull $\mathscr{D}_{trn}$ is the convex hull of set $\mathscr{D}_{trn}$. The following theorem states the performance of the neural network for a new input $\boldsymbol{\ell}^{\text{new}}$ that is not in $\mathscr{D}_{trn}$.

Theorem 7. *Given N input loads $\mathscr{D}_{trn} = \{\boldsymbol{\ell}^1, \ldots, \boldsymbol{\ell}^N\}$ and assume $\nabla_{\boldsymbol{\ell}} g_{\hat{\boldsymbol{\theta}}}(\boldsymbol{\ell}^i) = \boldsymbol{\mu}$ for all $i = 1, \ldots, N$. Assume the ICNN model $g_{\boldsymbol{\theta}}(\boldsymbol{\ell})$ is well-trained on $\mathscr{D}_{trn}$. Then for all points $\boldsymbol{\ell}^{new} \in$ convhull $\mathscr{D}_{trn}$, $\nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new}) = \boldsymbol{\mu}$.*

Proof. Since $\boldsymbol{\ell}^{\text{new}}$ is in the convex hull of $\mathscr{D}_{trn}$, there are positive coefficients $\alpha_1, \ldots, \alpha_N$ such that

$$\boldsymbol{\ell}^{\text{new}} = \alpha_1 \boldsymbol{\ell}^1 + \cdots + \alpha_N \boldsymbol{\ell}^N,$$

and $\alpha_1 + \ldots, \alpha_n = 1$. By convexity,

$$g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{\text{new}}) \leq \alpha_1 g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^1) + \cdots + \alpha_N g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^N). \tag{5.16}$$

By the assumption that $g_{\boldsymbol{\theta}}(\boldsymbol{\ell})$ is well-trained, we have

$$\nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^i) = \boldsymbol{\mu}, \text{ for } i = 1, \cdots, N. \tag{5.17}$$

Using first-order conditions of convex functions, we have

$$\begin{aligned} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{\text{new}}) &\geq g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^1) + \boldsymbol{\mu}(\boldsymbol{\ell}^{\text{new}} - \boldsymbol{\ell}^1) \\ &\cdots \\ g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{\text{new}}) &\geq g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^N) + \boldsymbol{\mu}(\boldsymbol{\ell}^{\text{new}} - \boldsymbol{\ell}^N). \end{aligned}$$

Multiplying the i'th equation by α_i and summing gives

$$g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{\text{new}}) \geq \alpha_1 g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^1) + \cdots + \alpha_N g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^N). \tag{5.19}$$

Combining (5.16) and (5.19) gives

$$g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{\text{new}}) = \alpha_1 g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^1) + \cdots + \alpha_N g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^N). \tag{5.20}$$

This implies the function is linear in the convex hull of $\mathscr{D}_{trn}$ and all the points have the same gradient. □

This theorem is useful for LP problems because the gradient of J is piecewise constant over convex polytopic regions. If we are given some points from a region, then a well-trained neural network guarantees that the function is learned correctly for all points within their convex hull. This result implies the correctness of the overall algorithms since they only rely on the gradient (dual variable) information.

Technically, the above theorem says that if the gradients of a convex function are equal at a set of points, then the function is linear on the convex hull of these points. This is not a surprising result, but it does show that by constraining the structure of the neural network and using gradient information, we can generalize to uncountable number of points (compact regions) by learning from a finite number of training points.

If training samples may not represent all possible regions, we can construct the KKT conditions augmented loss on a set of unlabeled data points, for which we only have access to the input load values but not the optimal cost and the optimal LMPs. Since we do not ask for labels, we can sample as many data points as we want. This allows us to train with a very large set. We can calculate the violations of KKT conditions, where the Lagrange multipliers are determined from the learned $\boldsymbol{\mu}$. For more detailed training process and KKT augmented loss, we refer to our additional work [14].

By training with the augmented dataset, unseen regions become "seen" in the sense that the outputs from the trained model must satisfy KKT conditions. As long as the model is well-trained, the analysis of generalization is the same as Section 5.6.3. Therefore, the generalization performance of training with helper set can also be guaranteed by Theorem 7.

5.6.4 Generalization for Unseen Regions

Now we consider the case where a region is **not represented at all** by the training data, neither the labeled nor the helper datasets. Then if a test sample comes from this region, would our method output anything useful? Methods like classification and dictionary learning cannot make useful predictions since there is no basis to make inferences for such unseen regions. The next theorem shows that our approach is still partially successful because the gradient of a test data point is bounded by the gradient of the training points:

Theorem 8. *Given N input loads $\mathscr{D}_{trn} = \{\boldsymbol{\ell}^1, \ldots, \boldsymbol{\ell}^N\}$, assume the ICNN model $g_{\boldsymbol{\theta}}(\boldsymbol{\ell})$ is well-trained on $\mathscr{D}_{trn}$. Assume that $N \geq n+1$ and $\mathscr{D}_{trn}$ does not lie in a lower dimensional subspace in $\mathbb{R}^n$. Then for all points $\boldsymbol{\ell}^{new} \in \text{convhull}\, \mathscr{D}_{trn}$, $\nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})$ is contained in a bounded convex polytope.*

Proof. Suppose $g_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}$ is a convex function. Given $\boldsymbol{\ell}^1, \ldots, \boldsymbol{\ell}^N$ in the domain of g and let $\boldsymbol{\mu}^i = \nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^i)$. Let $\boldsymbol{\ell}^{\text{new}}$ be a point in the convex hull of $\boldsymbol{\ell}^1, \ldots, \boldsymbol{\ell}^N$ and denote $\nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new}) = \boldsymbol{\mu}$. By the convexity of $g_{\boldsymbol{\theta}}$, we have

$$(\nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^i) - \nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new}))^T (\boldsymbol{\ell}^{\text{new}} - \boldsymbol{\ell}^i) \leq 0, \forall i. \tag{5.21}$$

The inequalities in (5.21) constrain the values that $\nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})$ can take. We are to show that these inequalities actually describe a closed and bounded polytope in $\mathbb{R}^n$. We do this by contradiction.

Suppose the region defined by the inequalities in (5.21) is not bounded. Then $\nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})$ can be scaled arbitrarily and all of the inequalities in (5.21) would still hold. Then we can take the norm of $\nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})$ to be large enough such that it would dominate the $\nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^i)$ terms. Then (5.21) becomes

$$\nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})^T (\boldsymbol{\ell}^{\text{new}} - \boldsymbol{\ell}^i) \geq 0, \forall i. \tag{5.22}$$

Since $\boldsymbol{\ell}^{\text{new}}$ is in the convex hull of $\boldsymbol{\ell}^1, \ldots, \boldsymbol{\ell}^N$, we can write it as $\boldsymbol{\ell}^{\text{new}} = \alpha_1 \boldsymbol{\ell}^1 + \cdots + \alpha_N \boldsymbol{\ell}^N$ and $\alpha_i \geq 0$ and sums up to 1. Substituting this into (5.22) and rearranging the terms, we have

$$\alpha_1 \nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})^T \boldsymbol{\ell}^1 + \cdots + \alpha_N \nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})^T \boldsymbol{\ell}^N \leq \nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})^T \boldsymbol{\ell}^i.$$

By the assumption that $N \geq n+1$ and $\boldsymbol{\ell}^1, \ldots, \boldsymbol{\ell}^N$ are not in a lower dimensional subspace of $\mathbb{R}^n$, $\nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})^T \boldsymbol{\ell}^i$ will be nonzero for at least two i's. But it is not possible to have a convex combination of scalars (the $\nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})^T \boldsymbol{\ell}^i$'s) larger than every scalar in the set when at least two are nonzero (this follows from Farkas' lemma). This contradicts the assumption that the polytope created by (5.21) is unbounded. □

The exact characterization of the polytope depends on the values of $\{\boldsymbol{\ell}^1, \ldots, \boldsymbol{\ell}^N\}$ and $\nabla_{\boldsymbol{\ell}} g_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})$. The significance of the theorem lies in that training data are able to constrain the gradient for all points that lies in its convex hull. Intuitively speaking, as long as some surrounding points are included in training, the gradient cannot be "very wrong" even for points coming from LP regions that were not seen during training.

Consider the curve in Fig 5.3. Suppose we learned the two end pieces correctly but there was no training data for the middle piece for the piecewise linear case. Then by Theorem 8, the slope of the middle piece is constrained to be between the slope of the two end pieces. Furthermore, even if the neural network is trained in such a way that there are more than one piece of the middle region, the slopes of all of the pieces are still bounded between the two end pieces. Since the proposed `Neural Decoder` only relies on getting μ to be in the correct range, the active constraints would be identified correctly for all of these cases.

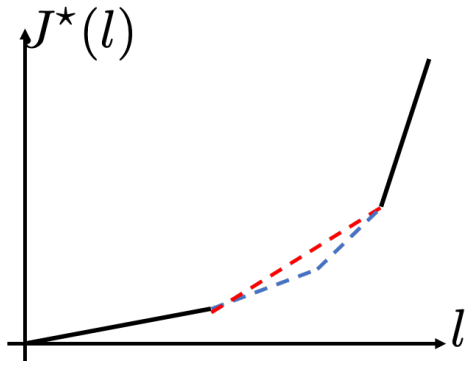

Figure 5.3: Example when the middle region has no data. But as long as the other two regions are well trained (black lines), the slopes in the middle region are bounded (blue and red dashed lines) by Theorem 8. Then the active constraint detection (Algorithm 1) would still be correct.

Theorem 8 formalizes the picture in Fig. 5.3 to higher dimensions, but the geometric intuition remains the same. This theorem also formalizes the empirical observation in [150], where the error of neural network-based OPF is reduced if training points are on the boundary of the feasible region.

5.7 Experimental Results

5.7.1 Experiment Setup

We evaluate the proposed learning approach, the *Neural Decoder*, on the IEEE 14-Bus and 39-Bus system [151] for both linear and quadratic costs. Specifically, over a wide range of problem input settings, we examine 1) solution quality in terms of constraint satisfaction and optimality and 2) computational efficiency over existing convex optimization solvers. Simulations are run on an unloaded Macbook Pro with Intel Core i5 8259U CPU @ 2.30GHz. The code and simulation data can be found at `https://github.com/chennnnnyize/Neural-Decoding-for-OPF`.

		80%-120%				20%-180%			
		NeuralDeco	E-to-E	Class	Neighbor	NeuralDeco	E-to-E	Class	Neighbor
14-Bus	Generator	98.49	96.36	94.64	89.03	93.19	90.39	71.37	73.65
	Line	99.91	96.06	98.52	92.44	93.47	93.77	94.07	84.33
	Infeasibility	0.77	3.36	6.63	11.28	4.03	13.69	24.62	29.19
39-Bus	Generator	99.91	93.30	98.95	93.40	93.95	44.38	90.19	63.90
	Line	99.91	94.85	99.24	59.68	93.48	72.22	86.28	71.34
	Infeasibility	0.02	11.07	0.01	29.38	4.25	61.25	13.49	38.71

Table 5.1: Performance comparison on linear cost case with different load input variations.

To generate the training set, we use CVXPY [152] powered by a CVXOPT solver [153] to solve (2.6). For each setting, we generate $\boldsymbol{\ell}$ by sampling from uniform distribution, with variations of 20%, and 80%. We solve $60,000$ data samples using CVXPY for each network model under each variance setting, and split 20% of the data as test samples.

We train and compare three other learning models in terms of finding optimal solutions while satisfying all constraints:

1. *Nearest neighbor for active constraints:* This benchmark is a sanity check on whether a deep neural network is needed or a simpler method would suffice. We fit and find a 3-nearest neighbor algorithm achieves the highest classification accuracy in predicting the active constraints.

2. *End-to-end regression:* Following [118], we construct a 4-layer neural network to fit the regression task of predicting optimal solution based on input $\boldsymbol{\ell}$. Mean squared error is used as training loss.

3. *Classification for active constraint sets:* Following [138], we construct a 4-layer neural network to predict the set of active constraints at optimal solution. We use one-hot encoding for different set of active constraints, and use cross-entropy as the loss function.

We term our framework the **Neural Decoder** and use a 3-layer neural network with 200 neurons on the first layer. We feed load vector $\boldsymbol{\ell}$ as the input for all of the methods. Once the active constraints are predicted, a linear equation solver is used to find final solutions. More simulation setup, architecture details, convergence analysis and ablation study can be found in the online version [13].

5.7.2 Simulation Results

Results on learning for the 14-bus and 39-bus OPF problem with linear costs and quadratic costs are listed in Table 5.1 and Table 5.2 respectively, where we report the accuracy of active generators' constraints and lines' constraints separately. Note that these simulation load samples are not sufficient to cover the input space and some test samples that do not reside in the same region as the training samples. Our method generalizes and has the lowest infeasibility across all test settings. For feasible test instances, the mean solution costs compared to optimal solutions provided by CVXPY is within 0.5%.

		80%-120%				20%-180%			
		NeuralDeco	E-to-E	Class	Neighbor	NeuralDeco	E-to-E	Class	Neighbor
14-Bus	Generator	93.43	95.39	89.73	92.44	94.19	78.68	68.71	65.18
	Line	98.61	93.81	98.85	92.90	81.01	72.55	67.51	74.51
	Infeasibility	2.12	6.01	4.07	5.82	8.18	31.87	17.81	29.64
39-Bus	Generator	97.95	96.87	98.12	97.99	85.08	36.71	29.06	14.28
	Line	99.08	99.18	99.45	98.97	84.32	42.82	56.18	21.07
	Infeasibility	0.38	2.84	0.42	1.14	11.63	73.52	58.62	78.92

Table 5.2: Performance comparison on quadratic cost case with different load input variations.

Feasibility and Optimality. Interestingly, our proposed method (NeuralDeco) can provide solutions with lower percentage of infeasibility than the sum of error on active generator and line constraints predictions. This is because identification steps of active generators and active lines are performed in sequence and we are making use of the information on total number of active constraints in the linear case, so the error on each step does not compound and lead to infeasible solutions.

The other methods perform much poorer. The nearest neighbor approach is not able to achieve good performance when the input variations are greater, showing that more sophisticated learning approaches are needed. The end-to-end prediction is hard to use for higher load variances, mainly because it does not explicitly include line flow constraints, so it tends to produce infeasible flow solutions. The classification approach is also not a proper learning strategy for larger-scale optimization problems, since the growing number of possible combination of active constraints leads to huge one-hot encodings at the classifier's output.

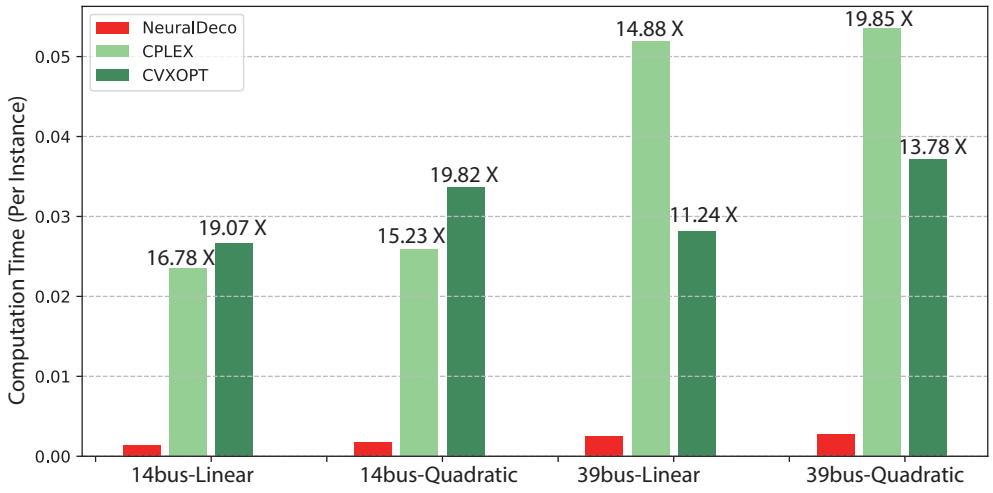

Figure 5.4: Average solving time per instance along with speedup statistics comparison for Neural Decoder, iterative solvers CVXOPT and CPLEX.

Computation Time. Compared to the solving process of optimization solvers, our proposed solver provides significant speed-up in all testing benchmarks as shown in Fig. 5.4. Compared to state-of-the-art commercial solvers, it can provide an order of magnitude in efficiency. We are using standard Python packages for neural network derivations and equation solvers, while further acceleration can be achieved by taking a batch of evaluating samples to calculate $\nabla_{\boldsymbol{\ell}} g_\theta(\boldsymbol{\ell})$, or adopting special linear equation solvers to solve the resulting sparse linear equations once all active constraints are identified. Please see [13] for more details and statistics, as well as just comparing the "core" computation times. That is, we discount the overhead in problem conversion, constraint translation and so on, and only record the time used by lower level linear algebra packages. The relative speeds remain virtually unchanged in this comparison from Fig. 5.4.

Price Forecasting. The results are shown in Fig.5.5 with forecasted prices and forecasting errors. For the case of 39-bus with 80% input variations, the overall LMP forecasting mean absolute percentage error (MAPE) on test samples is 2.15%. We also compare proposed method to a straightforward forecasting method, where we trained a neural network with nodal electricity demand as input and LMP as output, whose error is about 50% larger.

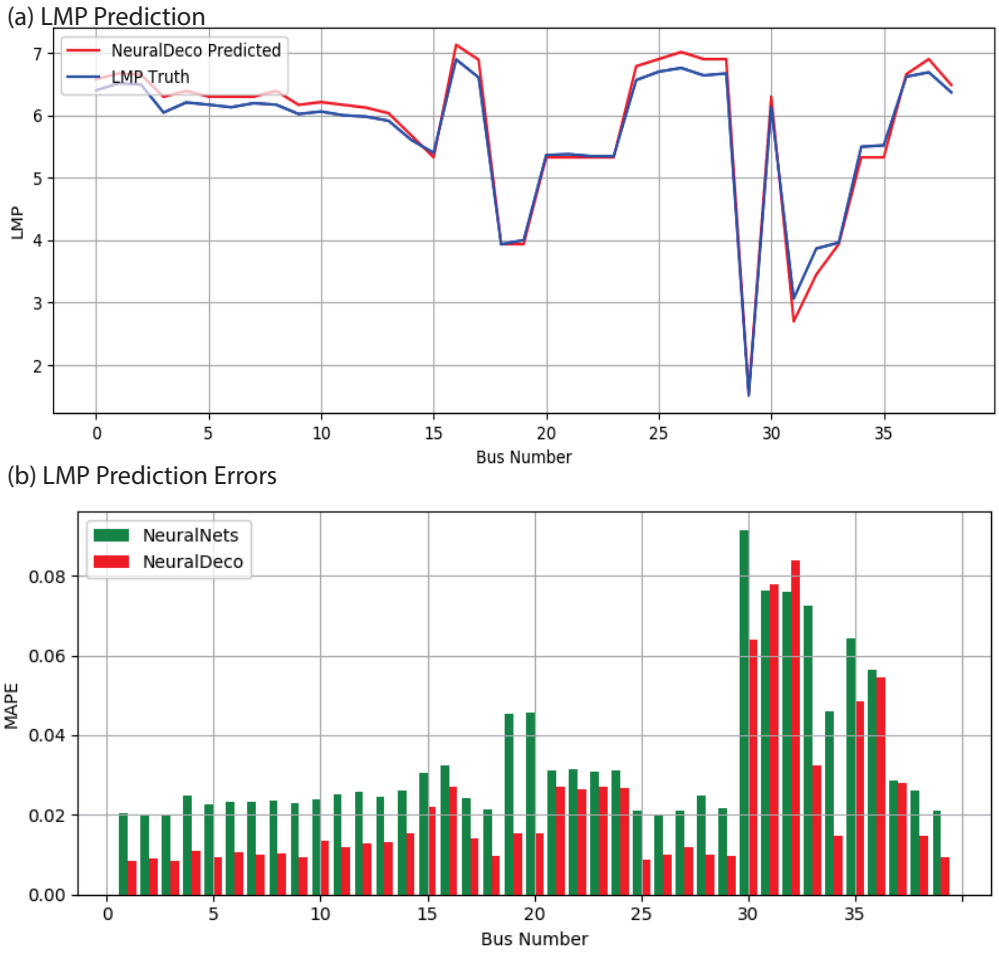

Figure 5.5: Simulation results on LMP forecasting on 39-bus system with 80% load variations. (a) LMPs of a single sample and (b). Average MAPE across all testing samples compared with neural network-based forecasts.

Generalization Results. To evaluate the generalization ability of our model, we create an illustrative example in the 14-bus system. Particularly, we examine the generalization performance on new data points that comes from region without any training samples. To generate the training set for this case, we only change the load values at two buses, but keep the remaining load values fixed. In this way, the space of input loads can be regarded as a two-dimensional plane. When varying the load values at the two buses, we can have four different combinations of active constraints, which correspond to four different values of $\boldsymbol{\mu}^\star$. Therefore, we divide the input load space as four regions,

denoted as R_0, R_1, R_2 and R_3. The division of the input space is shown in Fig. 5.6. We take training samples from R_0, R_2, and R_3, and take testing samples from R_1. Let us denote the training set as $\mathscr{D}_{trn}$, and the testing set as $\mathscr{D}_{tst}$.

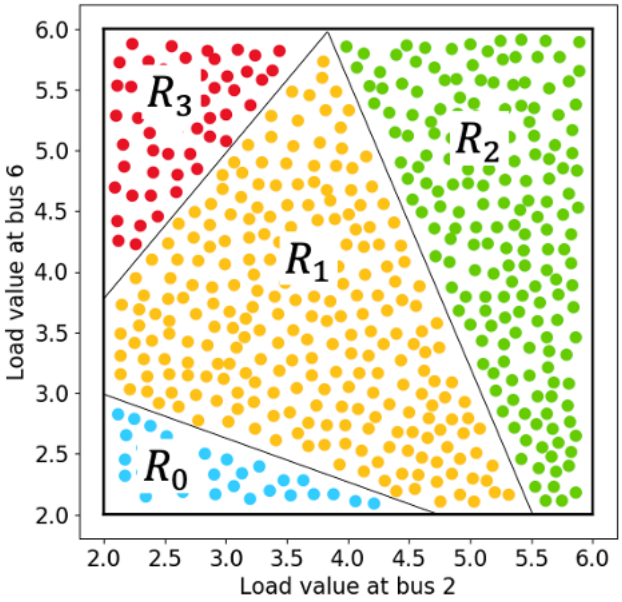

Figure 5.6: Division of the input space. The axes are load values at the two buses. In this example, based on different combinations of active constraints, the input load space can be divided into four regions. We take samples from R_1 as the testing set and samples from surrounding regions R_0, R_2 and R_3 as the training set.

We use two different training approaches for our model. In the first training approach, we only use $\mathscr{D}_{trn}$ for training and minimize both the regression loss and the KKT-related loss on $\mathscr{D}_{trn}$. For the second training approach, we construct an additional training set, called helper set $\mathscr{D}_{help}$. To generate the helper set, we can sample $\boldsymbol{\ell}$ from uniform distributions $\ell_2 \sim \mathsf{Uniform}(2,6)$ and $\ell_6 \sim \mathsf{Uniform}(2,6)$. Therefore, $\mathscr{D}_{help}$ contains the testing region R_1. Aside from minimizing the training loss on $\mathscr{D}_{trn}$, we also minimize the KKT-augmented loss which is further calibrated in [14] on $\mathscr{D}_{help}$. Note that $\mathscr{D}_{help}$ is not labeled. End-to-end model only use labeled samples and is trained on $\mathscr{D}_{trn}$.

We list the ratios of optimal, feasible and infeasible solutions obtained from different learning

	Optimality	Feasibility
End-to-end	5.52	8.6
Our model, with $\mathscr{D}_{\text{help}}$	**97.24**	**97.24**
Our model, without $\mathscr{D}_{\text{help}}$	**62.25**	**72.31**

Table 5.3: Generalization performance on test samples coming from never seen regions. With the helper set, our method is optimal 97% of the time. Even without the helper set, the optimality ratio is 62%. The end-to-end model fails to make reasonable predictions.

models in Table 5.3. As we can see, when we use $\mathscr{D}_{help}$ as an additional training set, we can obtain an optimality ratio as high as 96%. Even without $\mathscr{D}_{help}$, more than half of the solutions obtained from our model can achieve optimal values. As a comparison, the end-to-end model fails to make feasible predictions on test samples that come from never seen regions in the training process. The reason that our proposed algorithm outperforms the end-to-end model can be attributed to the KKT-related loss. By minimizing the KKT-related loss term, the trained model is able to learn the underlying KKT conditions in all four regions and make better predictions of $\boldsymbol{\mu}^\star$ on $\mathscr{D}_{tst}$.

5.8 Conclusion

This chapter laid the groundwork for learning to solve the optimal power flow problem using a decoding-style algorithm. By incorporating the knowledge of the optimization models and identifying the binding constraints at the optimal solution, we can build up our machine learning models upon the foundations of optimization theories. The algorithm described in this chapter can be treated as the building block for many "learning to optimize" problems.

An important future direction is to extend this work to ACOPF problems. For an ACOPF problem, if all the active constraints can be determined, the resulting AC power flow problem is still nonlinear, but in general simpler than the original optimization problem [154]. It would be important to compare against regression-based methods that learns a warm start to the ACOPF problem. For further extension, there are also great spaces in applying machine learning techniques

to solve integer programming problems, such as unit commitment, energy storage sitting and sizing problems, and etc. The evaluation of machine learning methods on computational efficiency, feasibility and optimality, and sample efficiency need to be further addressed.

Algorithm 4 Neural Decoder for Active Constraints

Input: $\boldsymbol{\ell}$, trained model g_θ, number of active constraints K

Input: (**Optional**) Dictionary $\{\tilde{\mathbf{A}}^T\boldsymbol{\mu}^*_{(k)}\}$, $\mathcal{P}_{\tilde{\mathbf{A}}^T\boldsymbol{\mu}^*_{(k)}}$, $\mathbf{f}_{(k)}$

Parameters: $\varepsilon > 0$, Optimization model $\tilde{\mathbf{A}}, \bar{\mathbf{f}}, \underline{\mathbf{f}}, \mathbf{c}$, $FLAG = 0$

Find $\nabla_{\boldsymbol{\ell}} g_\theta(\boldsymbol{\ell})$
for $i = 1, ..., n$ **do**
  if $\nabla_{l_i} g_\theta(\boldsymbol{\ell}) - c_i < -\varepsilon$ **then**
    $x_i = 0$, $K = K - 1$
  else if $\nabla_{l_i} g_\theta(\boldsymbol{\ell}) - c_i > \varepsilon$ **then**
    $x_i = \bar{x}_i$, $K = K - 1$
  end if
end for
for $j = 1, ..., |M|$ **do**
  $\boldsymbol{\delta}_{(j)} = \tilde{\mathbf{A}} \nabla_{\boldsymbol{\ell}} g_\theta(\boldsymbol{\ell}) - \tilde{\mathbf{A}}^T \boldsymbol{\mu}^*_{(j)}$
  if $\boldsymbol{\delta}_{(j)} \in \mathcal{P}_{\tilde{\mathbf{A}}^T\boldsymbol{\mu}^*_{(j)}}$ **then**
    Identify active flow constraints $\mathbf{f}_{(j)}$, $FLAG = 1$
    break
  end if
end for
if $FLAG = 0$ **then**
  `IHTSolve`((5.6), sparsity=K)
end if
`EquationSolve`($\mathbf{x} + \mathbf{A}\mathbf{f} = \boldsymbol{\ell}$, active constraints set)

Chapter 6

CONCLUSION AND FUTURE DIRECTIONS

6.1 Conclusion

We are working towards a world in which we have an increasingly better handle on how data-driven technologies can be integrated in existing and new critical infrastructures while safeguarding and representing important values such as safety, sustainability and equity. In this thesis we illustrated how theories from optimization and control can be utilized to realized engineering objectives. We have introduced new building blocks and algorithmic tools that can embed physical knowledge and optimization theories into machine learning and decision making pipelines. This thesis presents a set of computational methodologies faced with complex system and environment dynamics, and the combination of physical knowledge and optimization theory give the theoretical guarantees for the resulting algorithms.

In the context of energy and power systems, we proposed a pipeline by leveraging the availability of heterogeneous sources of sensing, actuation and data to enable high levels of penetration of renewable energy resources through a combination of uncertainty forecasting, optimal controller design and real-time decision-making. Such design techniques have the promises of providing safe, reliable and cost-effective services given our quickly evolving energy infrastructures.

This thesis has been a first step towards addressing the ambitious goals of achieving sustainable energy system. We utilized historical time series data to build the first generative models for renewables scenario generation, which provide a critical tool for power system planning and operation under environment uncertainty. We have shown by constructing an input convex neural network, we are able to bridge the optimal control theory with the complex yet unknown system dynamics. The resulting optimal control framework can strike the balance between computation tractability and model representation. In order to achieve real-time decision making for large scale

optimization problems, we set up a novel learning framework built upon on the optimization theories. Such data-driven optimization solvers can provide an order of magnitude acceleration compared to standard solvers in specific energy dispatch tasks.

6.2 Future Directions

We are currently witnessing a perfect storm in achieving sustainable energy systems with the advancements in fundamental technology, societal incentives and infrastructure revolution. In the following, we provide a brief outlook of how the integration of machine learning, optimization and control could contribute to the development of sustainable energy systems.

- **Stochastic optimization and end-to-end decision-making.** We are faced with a clean energy future with high level penetration of renewables and end-user demands. When the overall energy systems have objectives of sustainability and efficiency, it is important to come up with algorithms to incorporate both the knowledge of the stochastic environments and the modeling tools of physical systems. Such motivations open the door for developing machine learning algorithms that could take the data uncertainty or model uncertainty into account. One possible route is to explore differentiable architecture that is able to handle a family of optimization problems which is explored in [141].

- **Multi-agent systems and complex interactions.** The inevitable transition to power systems fueled by solely renewable energy sources is only getting started. When most of the energy is generated locally, the source of stability from large-scale centralized generation will deplete, and local networks will have to organize the control of a stable voltage signal by themselves, necessitating grid-forming rather than grid-following inverter and DER control architectures. Motivated by the exciting idea that local networks will have to become more self-sufficient, this transition may contribute to more equitable access and control over electric energy. There is a need to characterize the dynamics of such distributed, low-inertia systems, and there are open spaces for market mechanisms, policy making, distributed optimization and machine learning.

- **System security with heterogeneous resources.** With huge amount of distributed energy resources connected to the electric grid and growing number of sensing and user data rushing into control room and algorithms, there are growing concerns regarding power system security. Our previous work explored the possibilities of forecasting algorithm vulnerabilities [19, 20]. While more rigorous analysis is a necessity to evaluate system robustnss against perturbed data or malicious system states. It is also an exciting area to investigate the dynamic coupling between user privacy and system security.

BIBLIOGRAPHY

[1] P. Pinson, H. Madsen, H. A. Nielsen, G. Papaefthymiou, and B. Klöckl, "From probabilistic forecasts to statistical scenarios of short-term wind power production," *Wind energy*, vol. 12, no. 1, pp. 51–62, 2009.

[2] Y. Chen, Y. Wang, D. Kirschen, and B. Zhang, "Model-free renewable scenario generation using generative adversarial networks," *IEEE Transactions on Power Systems*, vol. 33, no. 3, pp. 3265–3275, 2018.

[3] Y. Chen, X. Wang, and B. Zhang, "An unsupervised deep learning approach for scenario forecasts," in *2018 Power Systems Computation Conference (PSCC)*, pp. 1–7, IEEE, 2018.

[4] Y. Chen, P. Li, and B. Zhang, "Bayesian renewables scenario generation via deep generative networks," in *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, IEEE, 2018.

[5] Y. Chen, M. U. Hashmi, D. Deka, and M. Chertkov, "Stochastic battery operations using deep neural networks," in *2019 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pp. 1–5, IEEE, 2019.

[6] J. E. Contreras-Ocana, Y. Chen, U. Siddiqi, and B. Zhang, "Non-wire alternatives: An additional value stream for distributed energy resources," *IEEE Transactions on Sustainable Energy*, vol. 11, no. 3, pp. 1287–1299, 2019.

[7] C. Jiang, Y. Chen, Y. Mao, Y. Chai, and M. Yu, "Forecasting spatio-temporal renewable scenarios: a deep generative approach," *arXiv preprint arXiv:1903.05274*, 2019.

[8] B. Amos, L. Xu, and J. Z. Kolter, "Input convex neural networks," in *International Conference on Machine Learning*, pp. 146–155, 2017.

[9] Y. Chen, Y. Shi, and B. Zhang, "Modeling and optimization of complex building energy systems with deep neural networks," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pp. 1368–1373, IEEE, 2017.

[10] Y. Chen, Y. Shi, and B. Zhang, "Optimal control via neural networks: A convex approach," in *International Conference on Learning Representations (ICLR)*, pp. 1–11, 2019.

[11] Y. Chen, Y. Shi, and B. Zhang, "Input convex neural networks for optimal voltage regulation," *arXiv preprint arXiv:2002.08684*, 2020.

[12] Y. Chen, Y. Shi, and B. Zhang, "Data-driven optimal voltage regulation using input convex neural networks," *Electric Power Systems Research*, vol. 189, p. 106741, 2020.

[13] Y. Chen and B. Zhang, "Learning to solve network flow problems via neural decoding," *arXiv:2002.04091*, 2020.

[14] L. Zhang, Y. Chen, and B. Zhang, "A convex neural network solver for dcopf with generalization guarantees," *arXiv preprint arXiv:2009.09109*, 2020.

[15] Y. Chen, W. Yang, and B. Zhang, "Using mobility for electrical load forecasting during the covid-19 pandemic," *arXiv preprint arXiv:2006.08826*, 2020.

[16] X.-W. Wang, Y. Chen, and Y.-Y. Liu, "Link prediction through deep generative model," *Iscience*, vol. 23, no. 10, p. 101626, 2020.

[17] Y. Chen, Q. Li, and H. Wang, "Towards trusted social networks with blockchain technology," in *Symposium on Foundations and Applications of Blockchain*, p. 37, 2018.

[18] H. Hosseini, Y. Chen, S. Kannan, B. Zhang, and R. Poovendran, "Blocking transferability of adversarial examples in black-box learning systems," *arXiv preprint arXiv:1703.04318*, 2017.

[19] Y. Chen, Y. Tan, and B. Zhang, "Exploiting vulnerabilities of load forecasting through adversarial attacks," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, pp. 1–11, 2019.

[20] Y. Chen, Y. Tan, and D. Deka, "Is machine learning in power systems vulnerable?," in *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 1–6, IEEE, 2018.

[21] W. Wolf, "Cyber-physical systems," *Computer*, vol. 42, pp. 88–89, 2009.

[22] C.-C. Cheng, S. Pouffary, N. Svenningsen, and J. M. Callaway, "The kyoto protocol, the clean development mechanism and the building and construction sector: A report for the unep sustainable buildings and construction initiative," 2008.

[23] Z. Zhang, R. Deng, T. Yuan, and S. J. Qin, "Distributed optimization of multi-building energy systems with spatially and temporally coupled constraints," in *American Control Conference (ACC), 2017*, pp. 2913–2918, IEEE, 2017.

[24] Y. Ma, A. Kelman, A. Daly, and F. Borrelli, "Predictive control for energy efficient buildings with thermal storage: Modeling, stimulation, and experiments," *IEEE Control Systems*, vol. 32, no. 1, pp. 44–64, 2012.

[25] P. H. Shaikh, N. B. M. Nor, P. Nallagownden, I. Elamvazuthi, and T. Ibrahim, "A review on optimized control systems for building energy and comfort management of smart sustainable buildings," *Renewable and Sustainable Energy Reviews*, vol. 34, pp. 409–429, 2014.

[26] D. Belanger, "Deep energy-based models for structured prediction," 2017.

[27] J.-L. Wu, H. Xiao, and E. Paterson, "Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework," *Physical Review Fluids*, vol. 3, no. 7, p. 074602, 2018.

[28] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[30] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[31] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[32] K. Zhou, J. C. Doyle, K. Glover, *et al.*, *Robust and optimal control*, vol. 40. Prentice hall New Jersey, 1996.

[33] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*, vol. 6. Athena Scientific Belmont, MA, 1997.

[34] S. Sra, S. Nowozin, and S. J. Wright, *Optimization for machine learning*. Mit Press, 2012.

[35] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033, IEEE, 2012.

[36] B. Zhang, A. Y. Lam, A. D. Domínguez-García, and D. Tse, "An optimal and distributed method for voltage regulation in power distribution systems," *IEEE Transactions on Power Systems*, vol. 30, no. 4, pp. 1714–1726, 2014.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[38] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

[39] M. E. Baran and F. F. Wu, "Optimal capacitor placement on radial distribution systems," *IEEE Transactions on power Delivery*, vol. 4, no. 1, pp. 725–734, 1989.

[40] R. Srikant and L. Ying, *Communication networks: an optimization, control, and stochastic networks perspective*. Cambridge University Press, 2013.

[41] J. L. Gross and J. Yellen, *Graph theory and its applications*. CRC press, 2005.

[42] H. Zhu and H. J. Liu, "Fast local voltage control under limited reactive power: Optimality and stability analysis," *IEEE Transactions on Power Systems*, vol. 31, no. 5, pp. 3794–3803, 2015.

[43] P. Li, B. Jin, D. Wang, and B. Zhang, "Distribution system voltage control under uncertainties using tractable chance constraints," *IEEE Transactions on Power Systems*, 2018.

[44] M. Farivar, R. Neal, C. Clarke, and S. Low, "Optimal inverter var control in distribution systems with high pv penetration," in *2012 IEEE Power and Energy Society general meeting*, pp. 1–7, IEEE, 2012.

[45] M. Farivar, C. R. Clarke, S. H. Low, and K. M. Chandy, "Inverter var control for distribution systems with renewables," in *2011 IEEE international conference on smart grid communications (SmartGridComm)*, pp. 457–462, IEEE, 2011.

[46] Q. Peng and S. Low, "Distributed algorithm for optimal power flow on unbalanced multiphase distribution networks," *arXiv preprint arXiv:1512.06482*, 2015.

[47] J. D. Watson, N. R. Watson, and I. Lestas, "Optimized dispatch of energy storage systems in unbalanced distribution networks," *IEEE Transactions on Sustainable Energy*, vol. 9, no. 2, pp. 639–650, 2017.

[48] Y. Liao, Y. Weng, G. Liu, Z. Zhao, C.-W. Tan, and R. Rajagopal, "Unbalanced three-phase distribution grid topology estimation and bus phase identification," *arXiv preprint arXiv:1809.07192*, 2018.

[49] H. Holttinen, P. Meibom, C. Ensslin, L. Hofmann, J. Mccann, J. Pierik, *et al.*, "Design and operation of power systems with large amounts of wind power," in *VTT Research Notes 2493*, Citeseer, 2009.

[50] M. R. Patel, *Wind and solar power systems: design, analysis, and operation*. CRC press, 2005.

[51] J. M. Morales, R. Minguez, and A. J. Conejo, "A methodology to generate statistically dependent wind speed scenarios," *Applied Energy*, vol. 87, no. 3, pp. 843–855, 2010.

[52] S. Delikaraoglou and P. Pinson, "High-quality wind power scenario forecasts for decision-making under uncertainty in power systems," in *13th International Workshop on Large-Scale Integration of Wind Power into Power Systems as well as on Transmission Networks for Offshore Wind Power (WIW 2014)*, 2014.

[53] W. B. Powell and S. Meisel, "Tutorial on stochastic optimization in energy part i: Modeling and policies," *IEEE Transactions on Power Systems*, vol. 31, no. 2, pp. 1459–1467, 2016.

[54] C. Zhao and Y. Guan, "Unified stochastic and robust unit commitment," *IEEE Transactions on Power Systems*, vol. 28, no. 3, pp. 3353–3361, 2013.

[55] Y. Wang, Y. Liu, and D. S. Kirschen, "Scenario reduction with submodular optimization," *IEEE Transactions on Power Systems*, vol. 32, no. 3, pp. 2479–2480, 2017.

[56] S. Mitra, "Scenario generation for stochastic programming," tech. rep., OPTIRISK systems, 2006.

[57] J. R. Birge and F. Louveaux, *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[58] R. Barth, L. Söder, C. Weber, H. Brand, and D. J. Swider, "Methodology of the scenario tree tool," *Wilmar Deliverable*, vol. 6, 2006.

[59] T. Wang, H.-D. Chiang, and R. Tanabe, "Toward a flexible scenario generation tool for stochastic renewable energy analysis," in *Power Systems Computation Conference (PSCC), 2016*, pp. 1–7, IEEE, 2016.

[60] X.-Y. Ma, Y.-Z. Sun, and H.-L. Fang, "Scenario generation of wind power based on statistical uncertainty and variability," *IEEE Transactions on Sustainable Energy*, vol. 4, no. 4, pp. 894–904, 2013.

[61] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[62] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[63] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *arXiv preprint arXiv:1701.07875*, 2017.

[64] C. Villani, *Optimal transport: old and new*, vol. 338. Springer Science & Business Media, 2008.

[65] C. Wan, Z. Xu, P. Pinson, Z. Y. Dong, and K. P. Wong, "Probabilistic forecasting of wind power generation using extreme learning machine," *IEEE Transactions on Power Systems*, vol. 29, no. 3, pp. 1033–1044, 2014.

[66] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147, 2013.

[67] C. Draxl, A. Clifton, B.-M. Hodge, and J. McCaa, "The wind integration national dataset (wind) toolkit," *Applied Energy*, vol. 151, pp. 355–366, 2015.

[68] P. Pinson and R. Girard, "Evaluating the quality of scenarios of short-term wind power generation," *Applied Energy*, vol. 96, pp. 12–20, 2012.

[69] N. K. Suryadevara, S. C. Mukhopadhyay, S. D. T. Kelly, and S. P. S. Gill, "Wsn-based smart sensors and actuator for power management in intelligent buildings," *IEEE/ASME transactions on mechatronics*, vol. 20, no. 2, pp. 564–571, 2015.

[70] D. Meger, J. C. G. Higuera, A. Xu, P. Giguere, and G. Dudek, "Learning legged swimming gaits from experience," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 2332–2338, IEEE, 2015.

[71] M. P. Deisenroth, C. E. Rasmussen, and D. Fox, "Learning to control a low-cost manipulator using data-efficient reinforcement learning," 2011.

[72] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, pp. 6000–6010, 2017.

[73] A. Magnani and S. P. Boyd, "Convex piecewise-linear fitting," *Optimization and Engineering*, vol. 10, no. 1, pp. 1–17, 2009.

[74] L. Ljung, "System identification," in *Signal analysis and prediction*, pp. 163–173, Springer, 1998.

[75] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566, IEEE, 2018.

[76] K. Kawaguchi, "Deep learning without poor local minima," in *Advances in Neural Information Processing Systems*, pp. 586–594, 2016.

[77] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[78] S. Levine and V. Koltun, "Learning complex neural network policies with trajectory optimization," in *International Conference on Machine Learning*, pp. 829–837, 2014.

[79] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[80] T. Wei, Y. Wang, and Q. Zhu, "Deep reinforcement learning for building hvac control," in *The Design and Automation Conference*, 2017.

[81] D. O'Neill, M. Levorato, A. Goldsmith, and U. Mitra, "Residential demand response using reinforcement learning," in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pp. 409–414, IEEE, 2010.

[82] A. Filos, P. Tigkas, R. McAllister, N. Rhinehart, S. Levine, and Y. Gal, "Can autonomous vehicles identify, recover from, and adapt to distribution shifts?," in *International Conference on Machine Learning*, pp. 3145–3153, PMLR, 2020.

[83] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *International conference on machine learning*, pp. 5331–5340, PMLR, 2019.

[84] M. Laskin, A. Srinivas, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," in *International Conference on Machine Learning*, pp. 5639–5650, PMLR, 2020.

[85] H. Xu, A. D. Domínguez-García, and P. W. Sauer, "A data-driven voltage control framework for power distribution systems," in *2018 IEEE Power & Energy Society General Meeting (PESGM)*, pp. 1–5, IEEE, 2018.

[86] Q. Yang, G. Wang, A. Sadeghi, G. B. Giannakis, and J. Sun, "Real-time voltage control using deep reinforcement learning," *arXiv preprint arXiv:1904.09374*, 2019.

[87] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*, vol. 2. MIT press Cambridge, 1998.

[88] S. Miryoosefi, K. Brantley, H. Daumé III, M. Dudik, and R. Schapire, "Reinforcement learning with convex constraints," in *Advances in Neural Information Processing Systems*, pp. 1–9, 2019.

[89] B. Hanin, "Universal function approximation by deep neural nets with bounded width and relu activations," *arXiv preprint arXiv:1708.02691*, 2017.

[90] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, "Understanding deep neural networks with rectified linear units," *arXiv preprint arXiv:1611.01491*, 2016.

[91] S. Skogestad and I. Postlethwaite, *Multivariable feedback control: analysis and design*, vol. 2. Wiley New York, 2007.

[92] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*, vol. 15. Siam, 1994.

[93] Y. Shi, B. Xu, Y. Tan, D. Kirschen, and B. Zhang, "Optimal battery control under cycle aging mechanisms in pay for performance settings," *IEEE Transactions on Automatic Control*, pp. 1–1, 2018.

[94] S. Kouro, P. Cortés, R. Vargas, U. Ammann, and J. Rodríguez, "Model predictive control—a simple and powerful method to control power converters," *IEEE Transactions on industrial electronics*, vol. 56, no. 6, pp. 1826–1838, 2009.

[95] P. M. Carvalho, P. F. Correia, and L. A. Ferreira, "Distributed reactive power generation control for voltage rise mitigation in distribution networks," *IEEE transactions on Power Systems*, vol. 23, no. 2, pp. 766–772, 2008.

[96] P. Jahangiri and D. C. Aliprantis, "Distributed volt/var control by pv inverters," *IEEE Transactions on power systems*, vol. 28, no. 3, pp. 3429–3439, 2013.

[97] M. E. Baran and F. F. Wu, "Network reconfiguration in distribution systems for loss reduction and load balancing," *IEEE Transactions on Power delivery*, vol. 4, no. 2, pp. 1401–1407, 1989.

[98] D. Deka, S. Backhaus, and M. Chertkov, "Estimating distribution grid topologies: A graphical learning based approach," in *2016 Power Systems Computation Conference (PSCC)*, pp. 1–7, IEEE, 2016.

[99] H. Li, Y. Weng, Y. Liao, B. Keel, and K. E. Brown, "Robust hidden topology identification in distribution systems," *arXiv preprint arXiv:1902.01365*, 2019.

[100] G. Qu and N. Li, "An optimal and distributed feedback voltage control under limited reactive power," in *2018 Power Systems Computation Conference (PSCC)*, pp. 1–7, IEEE, 2018.

[101] M. Grant and S. Boyd, "Cvx: Matlab software for disciplined convex programming, version 2.1," 2014.

[102] J. N. Tsitsiklis, "Problems in decentralized decision making and computation.," tech. rep., Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems, 1984.

[103] W. Yu and R. Lui, "Dual methods for nonconvex spectrum optimization of multicarrier systems," *IEEE Transactions on Communications*, vol. 54, no. 7, pp. 1310–1322, 2006.

[104] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, 2007.

[105] Z. Wen, D. O'Neill, and H. Maei, "Optimal demand response using device-based reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2312–2324, 2015.

[106] G. Dalal, E. Gilboa, and S. Mannor, "Hierarchical decision making in electricity grid management," in *International Conference on Machine Learning*, pp. 2197–2206, 2016.

[107] J. Jimenez, "How the Texas power grid braces against rolling blackouts as summer heat looms," *The Dallas Morning News*, 2019.

[108] I. Penn, "This did not go well: Inside PG&E's blackout control room," *The New York Times*, 2019.

[109] J. D. Glover, T. J. Overbye, and M. S. Sarma, *Power System Analysis and Design*. CENGAGE Learning, 2017.

[110] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on power systems*, vol. 26, no. 1, pp. 12–19, 2010.

[111] F. Milano, "An open source power system analysis toolbox," *IEEE Transactions on Power systems*, vol. 20, no. 3, pp. 1199–1206, 2005.

[112] B. Stott, J. Jardim, and O. Alsaç, "Dc power flow revisited," *IEEE Transactions on Power Systems*, vol. 24, no. 3, pp. 1290–1300, 2009.

[113] F. Li and R. Bo, "Dcopf-based lmp simulation: algorithm, comparison with acopf, and sensitivity," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 1475–1485, 2007.

[114] A. Hauswirth, S. Bolognani, G. Hug, and F. Dörfler, "Projected gradient descent on riemannian manifolds with applications to online power system optimization," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 225–232, IEEE, 2016.

[115] Y. Zhang, E. Dall'Anese, and M. Hong, "Dynamic admm for real-time optimal power flow," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1085–1089, IEEE, 2017.

[116] S. Huang and V. Dinavahi, "Fast batched solution for real-time optimal power flow with penetration of renewable energy," *IEEE Access*, vol. 6, pp. 13898–13910, 2018.

[117] W. Deng, Y. Ji, and L. Tong, "Probabilistic forecasting and simulation of electricity markets via online dictionary learning," *arXiv preprint arXiv:1606.07855*, 2016.

[118] X. Pan, T. Zhao, and M. Chen, "Deepopf: A deep neural network approach for security-constrained dc optimal power flow," *arXiv preprint arXiv:1910.14448*, 2019.

[119] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 136–145, JMLR. org, 2017.

[120] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," in *Advances in Neural Information Processing Systems*, pp. 9558–9570, 2019.

[121] D. Bertsimas, E. Litvinov, X. A. Sun, J. Zhao, and T. Zheng, "Adaptive robust optimization for the security constrained unit commitment problem," *IEEE transactions on power systems*, vol. 28, no. 1, pp. 52–63, 2012.

[122] R. E. Bixby, "Computational progress in linear and mixed integer programming," *Presentation at ICIAM*, vol. 2015, 2015.

[123] T. Heidel, F. Pan, S. Elbert, C. DeMarco, and H. Mittelmann, "Optimal power flow competition design considerations," in *Increasing Market and Planning Efficiency through Improved Software, FERC Technical Conference*, 2016.

[124] A. Gomez-Exposito, A. J. Conejo, and C. Canizares, *Electric energy systems: analysis and operation*. CRC press, 2018.

[125] G. Dantzig and D. R. Fulkerson, "On the max flow min cut theorem of networks," *Linear inequalities and related systems*, vol. 38, pp. 225–231, 2003.

[126] R. J. Vanderbei *et al.*, *Linear programming*. Springer, 2015.

[127] S. S. Rao, *Engineering optimization: theory and practice*. John Wiley & Sons, 2019.

[128] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 5, pp. 554–562, 1988.

[129] W. E. Lillo, M. H. Loh, S. Hui, and S. H. Zak, "On solving constrained optimization problems with neural networks: A penalty method approach," *IEEE Transactions on neural networks*, vol. 4, no. 6, pp. 931–940, 1993.

[130] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Advances in Neural Information Processing Systems*, pp. 6348–6358, 2017.

[131] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi, "Exact combinatorial optimization with graph convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 15554–15566, 2019.

[132] 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.

[133] X. Cao, R. Ma, L. Liu, H. Shi, Y. Cheng, and C. Sun, "A machine learning-based algorithm for joint scheduling and power control in wireless networks," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4308–4318, 2018.

[134] W. Cui, K. Shen, and W. Yu, "Spatial deep learning for wireless scheduling," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1248–1261, 2019.

[135] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for wireless resource management," in *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–6, IEEE, 2017.

[136] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[137] D. Deka and S. Misra, "Learning for dc-opf: Classifying active sets using neural nets," in *2019 IEEE Milan PowerTech*, pp. 1–6, IEEE, 2019.

[138] Y. Ng, S. Misra, L. A. Roald, and S. Backhaus, "Statistical learning for dc optimal power flow," in *2018 Power Systems Computation Conference (PSCC)*, pp. 1–7, IEEE, 2018.

[139] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pp. 50–56, 2016.

[140] J. Song, R. Lanka, A. Zhao, Y. Yue, and M. Ono, "Learning to search via retrospective imitation," *arXiv preprint arXiv:1804.00846*, 2018.

[141] P. L. Donti, B. Amos, and J. Z. Kolter, "Task-based end-to-end model learning in stochastic optimization," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5490–5500, 2017.

[142] J. Djolonga and A. Krause, "Differentiable learning of submodular models," in *Advances in Neural Information Processing Systems*, pp. 1013–1023, 2017.

[143] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665, 2019.

[144] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," *Journal of Fourier analysis and Applications*, vol. 14, no. 5-6, pp. 629–654, 2008.

[145] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov, "Hamming distance metric learning," in *Advances in neural information processing systems*, pp. 1061–1069, 2012.

[146] B. Jansen, J. De Jong, C. Roos, and T. Terlaky, "Sensitivity analysis in linear programming: just be careful!," *European Journal of Operational Research*, vol. 101, no. 1, pp. 15–28, 1997.

[147] F. J. Pineda, "Generalization of back-propagation to recurrent neural networks," *Physical review letters*, vol. 59, no. 19, p. 2229, 1987.

[148] A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang, "Learning polynomials with neural networks," in *International conference on machine learning*, pp. 1908–1916, 2014.

[149] Z. Zhu, Y. Li, and Y. Liang, "Learning and generalization in overparameterized neural networks, going beyond two layers," in *Advances in Neural Information Processing Systems 32*, pp. 6158–6169, 2019.

[150] A. Venzke, G. Qu, S. Low, and S. Chatzivasileiadis, "Learning optimal power flow: Worst-case guarantees for neural networks," *arXiv preprint arXiv:2006.11029*, 2020.

[151] T. Athay, R. Podmore, and S. Virmani, "A practical method for the direct analysis of transient stability," *IEEE Transactions on Power Apparatus and Systems*, no. 2, pp. 573–584, 1979.

[152] S. Diamond and S. Boyd, "Cvxpy: A python-embedded modeling language for convex optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2909–2913, 2016.

[153] M. S. Andersen, J. Dahl, and L. Vandenberghe, "Cvxopt: A python package for convex optimization, version 1.1. 6," *Available at cvxopt. org*, vol. 54, 2013.

[154] F. Capitanescu, "Critical review of recent advances and further developments needed in ac optimal power flow," *Electric Power Systems Research*, vol. 136, pp. 57–68, 2016.

VITA

Yize Chen received a B.S. degree from Chu Kochen College at Zhejiang University, Hangzhou, China in June, 2016. He has been working on a Doctorate degree in Electrical and Computer Engineering at the University of Washington since September of 2016. He was a research intern at Los Alamos National Laboratory in 2018, and a research intern at Microsoft Research in Redmond in 2020.