RESILIENCE ENHANCEMENT OF POST-DISASTER POWER DISTRIBUTION SYSTEMS USING DEEP REINFORCEMENT LEARNING

By

RAED AWADH M ALOTAIBI

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

DOCTOR OF PHILOSOPHY IN
ELECTRICAL AND COMPUTER ENGINEERING

2026

Oakland University
Rochester, Michigan

Doctoral Advisory Committee:

Mohamed A. Zohdy, Ph.D., Chair
Amanpreet Kaur, Ph.D.
Ali Alghamdi, Ph.D.
William Edwards, Ph.D.
Zeina Al-Salman, Ph.D.

First and foremost, this dissertation is dedicated to God, for guidance and strength.

To my parents, for your prayers, guidance, and unwavering support throughout every stage of my life.

To my small family—Mashael, Battal, and Hattan—thank you for your patience, love, and sacrifices. You are my greatest motivation and my constant source of strength.

To my brothers and sisters, for your encouragement and for standing by me through the challenges of this journey.

To my friends, for your support, kindness, and the moments of relief and laughter that helped me keep going.

ACKNOWLEDGMENTS

In the name of God, the Most Merciful, the Most Compassionate, who granted me guidance, patience, and strength throughout this journey. I sincerely thank my advisor, Prof. Mohamed Zohdy, for his continuous guidance, support, and encouragement. His thoughtful feedback, patience, and deep expertise shaped this work and helped me move forward through challenges.

I am also grateful to my dissertation committee—Prof. Amanpreet Kaur, Prof. Ali Alghamdi, Prof. William Edwards, and Prof. Zeina Al-Salman—for their valuable time, insightful comments, and constructive recommendations that improved the quality of this dissertation.

Finally, I extend my heartfelt thanks to my family for their prayers, sacrifices, and unwavering support throughout my Ph.D. journey.

Raed Alotaibi

ABSTRACT

RESILIENCE ENHANCEMENT OF POST-DISASTER POWER DISTRIBUTION SYSTEMS USING DEEP REINFORCEMENT LEARNING

by

Raed Awadh M Alotaibi

Adviser: Mohamed A Zohdy, Ph.D.

Weather-driven extreme events are placing growing stress on aging distribution infrastructure and increasingly threaten continuity of service for critical loads during prolonged outages. Microgrids can enhance resilience by transitioning to islanded operations and supplying prioritized loads with local distributed energy resources (DERs); however, post-disaster restoration remains challenging because operators must make coupled discrete–continuous decisions under tight resource and operating constraints.

This dissertation addressed this challenge by developing a parameterized deep reinforcement learning controller, PDQN-CLR, that targets priority-weighted restoration while enforcing operational feasibility under scarcity. The PDQN-CLR modeled restoration as a hybrid action in which a discrete operational category was selected and paired with a continuous parameter vector specifying the DER real and reactive power setpoints. The approach was evaluated in a closed-loop OpenDSS environment using the IEEE 123-node feeder configured as an islanded microgrid with five grid-forming battery energy storage systems and 17 prioritized critical loads over a 72-step (36-hour) horizon

with a 30-minute decision interval. Snapshot power-flow evaluation was performed at each step. Uncertainty was represented through capacity-factor derating under sufficient and scarce regimes, and each episode randomized the fault scenario, derating level, and initial state of charge.

This dissertation also introduced (i) an uncertainty-aware evaluation protocol based on capacity-factor derating; (ii) a three-tier priority-weighted reward to encode the critical-load hierarchy; and (iii) a DER-aware load service rule that reduced voltage-only overstatement in islanded operation. With sufficient resources, PDQN-CLR achieved a mean PCS of 0.94 versus 0.80 for a Greedy baseline, while maintaining a low constraint violation score (CVS) of approximately 0.009 in both sufficient and scarce regimes. The baseline produced substantially larger violation magnitudes (CVS ≈ 0.388–0.667), indicating more severe and/or more frequent exceedances of the constraints. These results indicate that parameterized deep reinforcement learning can improve priority-weighted restoration when capacity is available and preserve feasibility as a primary outcome when scarcity limits achievable restoration.

TABLE OF CONTENTS

TABLE OF CONTENTS—Continued

TABLE OF CONTENTS—Continued

TABLE OF CONTENTS—Continued

TABLE OF CONTENTS—Continued

TABLE OF CONTENTS—Continued

LIST OF TABLES

LIST OF TABLES—Continued

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADMS	Advanced Distribution Management System
ADN	Active Distribution Network(s)
AI	Artificial Intelligence
BESS	Battery Energy Storage System
CF	Capacity Factor
CLR	Critical Load Restoration
CMDP	Constrained Markov Decision Process
CPO	Constrained Policy Optimization
CPU	Central Processing Unit
CRL	Constrained Reinforcement Learning
CVS	Constraint Violation Score
DDPG	Deep Deterministic Policy Gradient
DDQN	Double Deep Q-Network
DER	Distributed Energy Resource(s)
DG	Distributed Generator(s)
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
DSS	Distribution System Simulator
EMA	Exponential Moving Average
ES	Energy Storage

FLISR	Fault Location, Isolation, and Service Restoration
GFM	Grid-forming
GPU	Graphics Processing Unit
HILP	High-impact, low-probability
HPC	High-Performance Computing
KLU	KLU sparse solver
LIHP	Low-impact, high-probability
MDP	Markov Decision Process
PCS	Priority-Weighted Critical Score
PDQN	Parameterized Deep Q-Network
PDQN-CLR	Parameterized Deep Q-Network for Critical Load Restoration
PER	Prioritized Experience Replay
PID	Proportional–Integral–Derivative
PPO	Proximal Policy Optimization
PV	Photovoltaic
RCPO	Reward Constrained Policy Optimization
RL	Reinforcement Learning
SAC	Soft Actor-Critic
SOC	State of Charge
SR	Service Rates
TD	Temporal-Difference

CHAPTER ONE

INTRODUCTION

Modern power distribution networks are evolving towards a decentralized energy management paradigm driven by Distributed Energy Resources (DERs). These include renewable energy sources, such as solar and wind generators, as well as battery energy storage systems, which enhance grid stability and sustainability. However, despite these advancements, power systems remain vulnerable to disruptive events, particularly in post-disaster scenarios. Natural disasters, such as hurricanes, earthquakes, wildfires, and ice storms, pose significant threats to the continuous operation of electrical infrastructure, leading to widespread outages that affect millions of customers and critical services. In the United States alone, power outages cost the U.S. economy approximately $150 billion annually [1], and according to the 2023 American Housing Survey, approximately one in four households experienced complete power loss at least once within a 12-month period [2].

The increasing frequency and severity of extreme weather events, largely driven by climate change, have highlighted the limitations of traditional power system planning and operation paradigms. Hurricane Helene, which made landfall as a category 4 storm in September 2024, exemplified the devastating potential of such events. It caused $78.7 billion in damages and became the deadliest hurricane to strike the U.S. mainland since Hurricane Katrina in 2005 [3]. While reliability has long been the cornerstone of power system design, there is a growing recognition that reliability alone is insufficient to address the challenges posed by high-impact low-probability (HILP) events. This

recognition has led to an emerging focus on power system resilience as a complementary paradigm that specifically addresses a system's ability to anticipate, withstand, and rapidly recover from major disruptions.

This chapter provides the foundational context for this dissertation by introducing the concept of power system resilience, discussing the distinction between resilience and reliability, and presenting the key engineering approaches used to enhance resilience. Furthermore, it introduces deep reinforcement learning as a promising computational approach for developing intelligent and adaptive control strategies for post-disaster power distribution system restoration.

1.1 Power System Resilience

The concept of resilience has gained significant attention across multiple disciplines, from ecology and psychology to engineering and urban planning. In the context of power systems, resilience has become a critical consideration as the grid faces unprecedented challenges from both natural and human-induced threats. This section establishes the theoretical foundation for understanding power system resilience by defining the concept, distinguishing it from the related concept of reliability, and introducing the key dimensions that characterize a resilient power system.

1.1.1 Definition of Power System Resilience

The distinction between resilience and reliability is fundamental to understanding power system operations and planning. Reliability focuses on low-impact, high-probability (LIHP) events and ensures continuous service under expected conditions, measured by indices like SAIDI and SAIFI [4]. In contrast, resilience concerns the system’s ability to maintain or restore critical functions during high-impact, low-

probability (HILP) events. While power systems have traditionally prioritized reliability, the growing frequency of extreme weather, wildfires, and cyberattacks has highlighted the need for more adaptive and recovery-oriented planning.

Table 1.1: Comparison of Reliability and Resilience in Power Systems

Aspect	Reliability	Resilience
Event Type	Low-impact, high-probability	High-impact, low-probability
Planning Horizon	Long-term adequacy	Both short and long-term
Focus	Prevent service interruptions	Minimize impact and speed recovery
Metrics	SAIDI, SAIFI, CAIDI	Restoration time, critical load served
Analysis Approach	Probabilistic, historical data	Scenario-based, extreme events
Primary Concern	Steady-state performance	Dynamic response and adaptation

Resilience is interpreted differently across various disciplines. Scientific literature contains more than 70 definitions for the term resilience, ranging from a perspective that focuses on recovery to previous levels of performance following stress to a more comprehensive perspective that highlights adaptation or even transformation in response to unfamiliar and extreme disruptions [5]. In power systems, a practical framing treats resilience as the capability to anticipate rare, high-impact disturbances, limit performance degradation as the event unfolds, recover services rapidly, and incorporate lessons that reduce vulnerability to future events [4]. This interpretation aligns naturally with an event timeline—preparedness and resistance before the disturbance, resourcefulness and adaptation during degraded operation, and response and recovery afterward—often illustrated through a resilience curve that explicitly links system performance to the restoration time. Because operational decisions across these phases can involve complex

and high-dimensional system conditions, learning-based control has become attractive. For example, deep reinforcement learning has shown that a Deep Q-network can learn effective control policies directly from high-dimensional observations, supported by stabilizing mechanisms, such as experience replay and a separate target network [6].

To make the concept of resilience actionable, many studies adopt the 4Rs—robustness, redundancy, resourcefulness, and rapidity—originally articulated in the broader resilience literature and later widely used in power-system resilience discussions as a compact way to organize assessment and improvement priorities [7]. These dimensions provide a structured basis for interpreting how the grid should resist disruption, maintain service through alternatives, mobilize effective responses, and restore performance quickly.

Table 1.2: The 4Rs of Power System Resilience

Dimension	Description
Robustness	The strength to withstand disturbances without losing functionality. In power systems, this includes the physical integrity of infrastructure and the ability to maintain acceptable performance levels under stress.
Redundancy	Backup systems and alternate paths to maintain service during disruptions. This encompasses spare capacity, parallel feeders, tie switches, and distributed generation that can provide alternative supply routes.
Resourcefulness	The ability to mobilize people, systems, and solutions to respond to challenges. This includes intelligent decision-making capabilities, adaptive control systems, and effective coordination of response resources.

Dimension	Description
Rapidity	Speed of response and recovery to minimize impact and downtime. This measures how quickly the system can detect problems, isolate faults, and restore service to affected customers.

These four dimensions are interconnected and mutually reinforce each other. For example, investments in robustness (e.g., underground cabling) reduce the initial damage from an event, while improvements in resourcefulness (e.g., DRL-based control systems) enhance the system's ability to adapt and recover. Together, the 4Rs provide a comprehensive framework for evaluating and improving the resilience of power systems across all phases of the disaster cycle.

1.1.2 Resilience Engineering Approaches in Power Systems

Resilience engineering in power systems typically involves two complementary strategies, as categorized in [4]:

- **Hardening Measures**: Physical infrastructure reinforcements, such as underground cabling, flood-resistant substations, and reinforced poles that reduce exposure to damage. These are preventive but often expensive and limited in flexibility.
- **Smart/Operational Measures**: Adaptive technologies like Advanced Distribution Management Systems (ADMS), FLISR automation, microgrids, and DER coordination. These leverage real-time data, automation, and intelligent control to respond dynamically to disruptions.

This dissertation focuses on smart/operational measures, particularly the use of deep reinforcement learning to enable intelligent control and rapid post-disaster restoration.

1.1.3 Deep Reinforcement Learning for Resilient Power and Energy Systems

Among the emerging artificial intelligence techniques, Deep Reinforcement Learning (DRL) is uniquely suited to address the dynamic challenges of power system resilience [8]. Unlike traditional optimization methods, which can be computationally intensive and inflexible in the face of unexpected contingencies, DRL enables autonomous sequential decision-making in uncertain environments [9]. This capability is especially valuable during post-disaster recovery, when the grid must rapidly adapt to evolving damage and fluctuating operating conditions without relying on static, predefined rules.

By learning directly from interactions with the environment, DRL agents can optimize complex restoration tasks, such as topology reconfiguration and load prioritization, significantly faster than conventional solvers. Although the technical foundations of DRL are detailed in Section 2.2, its use here signals a shift from static planning to adaptive, learning-driven control. Nonetheless, challenges remain in applying DRL to large-scale systems, particularly in ensuring scalability and enforcing operational constraints, which this dissertation aims to address.

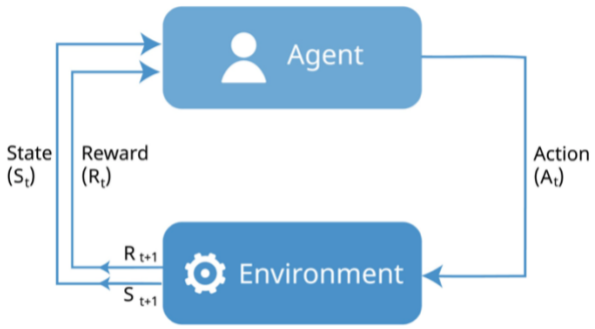

Figure 1.1: Reinforcement learning architecture diagram

The following section presents a comprehensive literature review of DRL applications in post-disaster power system restoration, examining the main objectives, agent roles, and operational modes considered in recent research, and identifying the gaps that motivate this dissertation's contributions.

1.2 Literature Review

Section 1.2 focuses specifically on the use of Deep Reinforcement Learning (DRL) for post-disaster energy management in power distribution systems. Building on the general background presented in the previous chapter, the aim here is to explore how DRL contributes to enhancing system resilience, ensuring critical load recovery, and optimizing restoration strategies after disruptive events. The review is organized into four key subsections: the main objectives addressed in post-disaster contexts, the functional roles of DRL-based agents, the operational modes considered during restoration, and the architectures and algorithms employed. It concludes with comparative tables that synthesize the literature across single-agent and multi-agent approaches.

1.2.1 Main Objectives in Post-Disaster DRL Research

The application of Deep Reinforcement Learning (DRL) in post-disaster energy management is typically driven by several core objectives that directly influence the resilience and recovery capabilities of power distribution systems. These objectives range from enhancing the speed and efficiency of system restoration to the prioritization of critical load recovery, cost-effective fault handling, and maintaining operational reliability. This subsection explores these primary goals by reviewing relevant literature and grouping the objectives into four distinct areas: distribution system restoration, load restoration, critical load prioritization, and fault recovery optimization.

1.2.1.1 Distribution System Restoration and Network Reconfiguration

A substantial body of DRL-based research has focused on the comprehensive restoration of distribution systems following large-scale disruptions. This research includes methods that integrate microgrid formation, switching operations, and distributed energy resource (DER) dispatch to reconfigure the grid dynamically and maximize load restoration. For example, studies such as [10] and [11] propose dynamic microgrid formation frameworks using Deep Q-networks to adaptively reconfigure the network based on outage patterns. These approaches model restoration as a Markov Decision Process (MDP) and leverage DRL to learn optimal switching actions in real time, ensuring radiality and operational feasibility. Similarly, [12] introduces a hierarchical scheme combining soft actor-critic (SAC) agents with quadratic programming, enabling coordinated operation across zones using integrated hybrid resources. Other works, such as [13] and [14], extend these ideas by incorporating expert demonstrations or hurricane scenario modeling to improve the adaptability and foresight of DRL agents. These models allow agents to anticipate fault propagation and prioritize service restoration accordingly. Meanwhile, [15] leverages graph reinforcement learning to perform real-time reconfiguration and load shedding by representing the distribution system as a graph and learning control policies directly from the network structure.

1.2.1.2 Load Restoration and Energy Management

Post-disaster load restoration represents a core objective in resilient distribution system operations, where the focus is on reconnecting and supplying as much of the disconnected demand as possible using local resources. In contrast to broader system reconfiguration, this task emphasizes load pick-up strategies that account for network

constraints, resource availability, and load criticality, often under challenging operational conditions. For instance, in [16] and [17], DRL agents learn sequential restoration policies involving both load pick-up and switching decisions under uncertain post-disaster conditions. In contrast, paper [18] uses one discrete-action DQN agent for prioritized load shedding and one continuous-action DDPG agent for DER dispatch — focusing on real-time decision-making under uncertainty rather than switching operations. Other works focus on enhancing resilience through improved generation dispatch directly tied to load restoration, as in [19], where multi-agent DDPG is used to coordinate DGs to minimize power mismatches and improve restoration rates. In [20], the restoration task is embedded within active distribution networks (ADNs), where coordination between DERs and loads requires simultaneous voltage control and energy balancing. Notably, [21] introduces a robust DRL framework that addresses the complexity of asynchronous and partial information during restoration, a scenario common in real-world post-disaster operations. Their method incorporates Monte Carlo Tree Search into the DRL training process, improving decision-making under limited observability and enabling large-scale restoration across thousands of buses.

1.2.1.3 Critical Load Restoration and Load Prioritization

Critical load restoration (CLR) focuses on prioritizing essential loads in post-disaster scenarios where resources are limited, and uncertainty is high. Recent advances apply DRL to optimize CLR decisions by dynamically coordinating DERs under system constraints. For instance, [22] employs a First-Order Meta-RL framework that enables the agent to quickly adapt restoration strategies across varying post-disaster scenarios by leveraging knowledge from previously encountered tasks, thereby enhancing

generalization and responsiveness under uncertainty. On the other hand, [23] models CLR as a partially observable problem to handle the uncertainties in renewable DER generation. It introduces a memory-based graph reinforcement learning approach, where the agent leverages both graph-structured data and temporal memory to improve decision-making under uncertainty. Similarly, [24] adopts a curriculum-based training approach to enhance the DRL agent's ability to manage renewable forecast errors, improving learning stability and restoration performance in uncertain post-disaster environments. These approaches demonstrate DRL's strength in handling sequential decisions, uncertainty, and nonlinear system dynamics more effectively than traditional methods.

1.2.1.4 Operational Resilience Support Functions

In addition to core restoration tasks, recent DRL research has explored operational resilience support functions—including voltage regulation, cost-aware dispatch, and resource coordination—to enhance system survivability and performance during and after disruptive events. For example, [25] utilizes a graph-enhanced deep reinforcement learning approach to perform autonomous fault recovery and load restoration in distribution networks, effectively learning topological relationships to minimize outage duration and recovery cost. Likewise, [26] proposes a hybrid-policy deep reinforcement learning algorithm integrated with robust optimization for post-disaster emergency dispatching in multi-park integrated energy systems. It combines discrete load-shedding decisions and continuous energy storage control to maximize energy supply revenue under uncertainty. In a broader context, [27] and [28] extend the DRL framework to microgrid-based systems that integrate cost-efficient DER scheduling with critical load

restoration. In [27], a dual-agent structure is adopted for networked microgrids, where separate agents manage normal and emergency operations to jointly minimize operational cost and enhance resilience. In contrast, [28] employs a single-agent Bayesian DRL approach for multi-energy microgrids, enabling unified coordination of DERs across coupled energy systems while maintaining cost efficiency and robust performance under uncertainty.

In addition to the objectives above, other resilience objectives have been addressed using DRL, such as maintaining voltage stability. For example, [29] proposes a multi-agent hybrid soft actor-critic framework that controls shunt reactive power compensators to mitigate voltage violations during and after extreme events. This approach enhances system survivability without relying on precise system models, demonstrating the flexibility of DRL in voltage regulation under multiple contingencies.

1.2.2 Agent Roles in DRL-Based Restoration

In post-disaster energy management, DRL agents are designed to perform specific control actions that directly influence the speed and efficiency of grid restoration. These agent roles vary depending on system objectives, available infrastructure, and the level of granularity required in decision-making. This section categorizes the primary functional roles of DRL-based agents into three key areas: dispatching power sources, such as distributed generators and energy storage systems; operating network reconfiguration elements, like switches and circuit breakers; and managing load dynamics through shedding, allocation, or prioritization. Each role reflects a critical aspect of autonomous system recovery and resilience enhancement.

1.2.2.1 Power Source Dispatch

A significant function of DRL-based agents in post-disaster scenarios is the autonomous dispatch of power sources, particularly DERs and DGs. These agents are responsible for real-time decisions regarding the coordination and utilization of generation assets to efficiently restore service. In various studies, such as [22] [25], , and [27], agents are explicitly designed to dispatch DERs, either as standalone controllers or as part of dual-agent frameworks that separate responsibilities between normal and emergency operations. In more complex configurations, like [12] and [20], agents manage DER dispatch in coordination with load and switching controls, enabling integrated energy balancing across active distribution networks or hierarchical zones. Other works, including [13], [14], [18], and [19], assign this role to single or distributed agents to manage generation-side control in alignment with load priorities. These agent roles are central to enhancing system resilience, ensuring local energy availability, and improving the speed and quality of post-disaster recovery efforts.

1.2.2.2 Network Reconfiguration

Another critical role assigned to DRL agents in post-disaster scenarios involves controlling network topology through the operation of switches and circuit breakers. These actions are essential for grid reconfiguration, fault isolation, and microgrid formation, which enable targeted restoration and load recovery. In studies, such as [10], [11], [12], [15], [20], and [25], agents are designed to operate sectionalizing or tie-line switches, often in coordination with DER dispatch or load management to maintain radiality and system feasibility during restoration. These switching actions may be performed by a dedicated agent or integrated within a broader control architecture. In

addition, circuit breaker operation is handled in [16] and [17], where agents use breaker control to disconnect or isolate faulted zones, thereby supporting protection schemes and minimizing service interruption. The ability to autonomously operate these components underscores the versatility of DRL agents in enabling adaptive, real-time reconfiguration under post-disaster constraints.

1.2.2.3 Load Management

Load management represents a vital function of DRL agents during post-disaster recovery, particularly when energy supply is limited, and prioritization is essential. Agents tasked with load shedding, allocation, or pick-up decisions enable targeted recovery strategies that optimize available resources and maintain critical services. In studies, such as [15], [18], , and [28] agents autonomously perform load shedding to relieve system stress or reallocate energy under constrained conditions. Some frameworks, like the one used in [27], incorporate both load allocation and shedding within a dual-agent architecture, allowing coordinated actions between normal and emergency operation modes. Additionally, [13] highlights the agent's role in managing load pick-up in tandem with DER dispatch, enabling fine-grained restoration sequencing based on system state and load criticality. These roles support adaptive, priority-based recovery and demonstrate how DRL agents can navigate complex trade-offs between service continuity and system stability.

1.2.3 Operational Modes

The operation mode of DRL-based control frameworks in post-disaster scenarios significantly influences agent design and decision-making complexity. Most studies operate within an islanded mode, where the distribution system is disconnected from the

main grid and relies solely on local resources, such as DERs and energy storage, for recovery and operation. This mode poses challenges related to resource scarcity, voltage stability, and autonomous coordination, making it a critical focus for resilience enhancement. However, a few works adopt grid-connected or hybrid modes, where the system operates normally in grid-connected conditions and transitions to islanded mode during emergencies. These hybrid frameworks allow agents to adapt between operational contexts, balancing cost optimization and resilience depending on the grid status. These operational distinctions are summarized in Tables 1.3 and 1.4, which highlights how restoration objectives and system assumptions shape agent design across the literature.

1.2.4 Summary of Reviewed Approaches

To conclude this chapter, the following tables summarize the key features of the reviewed studies, categorized by single-agent and multi-agent architectures. This structured comparison highlights the differences in agent roles, control strategies, and DRL design choices across the literature.

Table 1.3: Summary of Single-Agent DRL-Based Post-Disaster Restoration Approaches

Study	Objective(s)	Agent Role	Operation Mode	DRL Algorithm
[11]	Post-disaster Distribution System Restoration	Controls switches	Islanded	CM-DDQN
[13]	Post-disaster Distribution System Restoration	Controls DER dispatch and load pick-up decisions	Islanded	DDPG from Demonstrations
[14]	Post-disaster Distribution System Restoration	Dispatches DERs	Islanded	DDPG

Study	Objective(s)	Agent Role	Operation Mode	DRL Algorithm
[15]	Real-time Post-disaster Distribution System Restoration	Controls switches and load shedding	Both islanded and grid-connected	PPO
[21]	Post-disaster load restoration under asynchronous information conditions	Controls switches, DERs, and microgrid connections.	Islanded	DQL+ MCTS
[22]	Post-disaster critical load restoration	Dispatches DERs	Islanded	FOM-RL (ES-RL)
[23]	Post-disaster critical load restoration with uncertain DERs	Controls switches	Islanded	RGSAC
[24]	Post-disaster critical load restoration with uncertain renewable DERs	Dispatches DERs	Islanded	ES-RL + PPO
[28]	Cost optimization in normal conditions + post-disaster fault recovery: critical load restoration + cost optimization	Controls generators, gas storage, and load shedding	Both islanded and grid-connected	BDDPG

Table 1.4:Summary of Multi-Agent DRL-Based Post-Disaster Restoration Approaches

Study	Objective(s)	Agent Role	Operation Mode	DRL Algorithm
[10]	Post-disaster Distribution System Restoration	Controls switches	Islanded	DQN

Study	Objective(s)	Agent Role	Operation Mode	DRL Algorithm
[12]	Post-disaster Distribution System Restoration	Local agents: Dispatches DERs central CRC agent: Controls switches	Islanded	SAC + DDPG
[16]	Post-disaster load restoration	Controls Circuit Breakers	Islanded	MADQN
[17]	Post-disaster load restoration	Controls Circuit Breakers	Islanded	DQN
[18]	Post-disaster load restoration	Dual-agent control: • Load Agent: Controls load shedding • DGs Agent: Controls dispatchable generators (DGs)	Islanded	• Load Agent: DQN • DGs Agent: DDPG
[19]	Post-disaster load restoration	Dispatches DGs	Islanded	MADDPG
[20]	Post-disaster load restoration in Active Distribution Networks (ADNs)	Controls DERs and load switches Agent 1: Controls dispatchable generators (DGs) Agent 2: Controls energy storage systems (ESs) Agent 3: Controls	Islanded	MAGDPG

Study	Objective(s)	Agent Role	Operation Mode	DRL Algorithm
		load restoration decisions		
[25]	Post-disaster fault recovery (load restoration and cost optimization)	NRA: Controls switches PSA: Dispatches DERs	Islanded	MADDPG + GCN
[26]	Post-disaster fault recovery (load restoration and cost optimization)	Controls energy storage dispatch and load-shedding	Islanded	(HP-DDPG)
[27]	Cost optimization in normal conditions and Post-disaster fault recovery: critical load restoration and cost optimization	Dual-agent: • NOA (Normal Operating Agent): Dispatches DERs • EOA (Emergency Operating Agent): Allocates and sheds loads	Grid-connected - normal operation. Islanded - emergencies.	DDPG
[29]	Post-disaster voltage regulation	Controls the shunt reactive power compensators	Grid-connected	HSAC

1.2.5 Research Gaps

Based on the literature review, four key gaps were identified.

Gap 1: Hybrid Action Spaces Underexplored.

Most reinforcement learning (RL) approaches in power system control use either discrete (DQN-based) or continuous (DDPG/SAC) action space. However, DER dispatch

involves both discrete decisions (i.e., which units to activate) and continuous control (i.e., power levels). This hybrid action space structure is largely unaddressed in existing DRL applications.

Gap 2: Fixed DER Capacity Assumptions.

Current critical load restoration (CLR) studies often assume that DERs are fully operational post-disaster, neglecting capacity degradation due to component damage, fuel limitations, or partial failures. This assumption limits the realism and robustness of the DRL training and evaluation.

Gap 3: Uniform Load Treatment in Reward Design.

Many DRL methods apply uniform penalties or rewards across all loads, failing to reflect the real-world importance of load prioritization (e.g., critical infrastructure versus nonessential demand). Reward functions rarely incorporate a structured prioritization framework.

Gap 4: Voltage-Only Service Evaluation.

The load service status is commonly evaluated using voltage magnitude thresholds. However, in islanded microgrids with weak sources or unbalanced configurations, doing so may overestimate the actual load delivery, as the presence of voltage does not guarantee DER support. This approach introduces performance misrepresentation in both the training and evaluation phases.

1.3 Research Objectives

In response to the identified research gaps, this dissertation develops and evaluates PDQN-CLR, a PDQN-based controller for intelligent critical load restoration in islanded microgrids, building on the parameterized Deep Q-network framework for

hybrid discrete–continuous actions. The framework models DER dispatch using hybrid discrete-continuous actions, accounts for resource uncertainty, and embeds a three-tier priority structure into the control policy. To achieve this goal, the specific research objectives are to:

- Design a hybrid action space model that captures both discrete DER selection and continuous power dispatch decisions within a single architecture.
- Develop a DRL-based controller capable of operating robustly under both resource-scarce and resource-sufficient scenarios, considering randomized DER degradation and storage uncertainty.
- Incorporate a priority-weighted reward formulation to strictly guide the agent toward critical load restoration (e.g., hospitals and emergency services) under constrained conditions.
- Implement a robust load service evaluation mechanism that verifies true power delivery support from DERs to specific zones, eliminating false positives from voltage-only criteria.
- Evaluate policy generalization by validating the trained agent across multiple distinct disaster scenarios using the IEEE 123-bus distribution system.

1.4 Dissertation Organization

This dissertation is organized into six chapters. Chapter One introduces the research motivation, reviews related literature, identifies research gaps, and defines the problem and objectives. Then, Chapter Tow presents the theoretical background on critical load restoration and reinforcement learning, including hybrid and constrained formulations. Chapter Three describes the system model, disaster scenarios, and resource

configurations. Chapter Four details the proposed PDQN-based methodology for critical load restoration. Chapter Five presents experimental results and performance evaluation under multiple post-disaster scenarios. Finally, Chapter Six summarizes the key findings, discusses contributions and limitations, and outlines directions for future research.

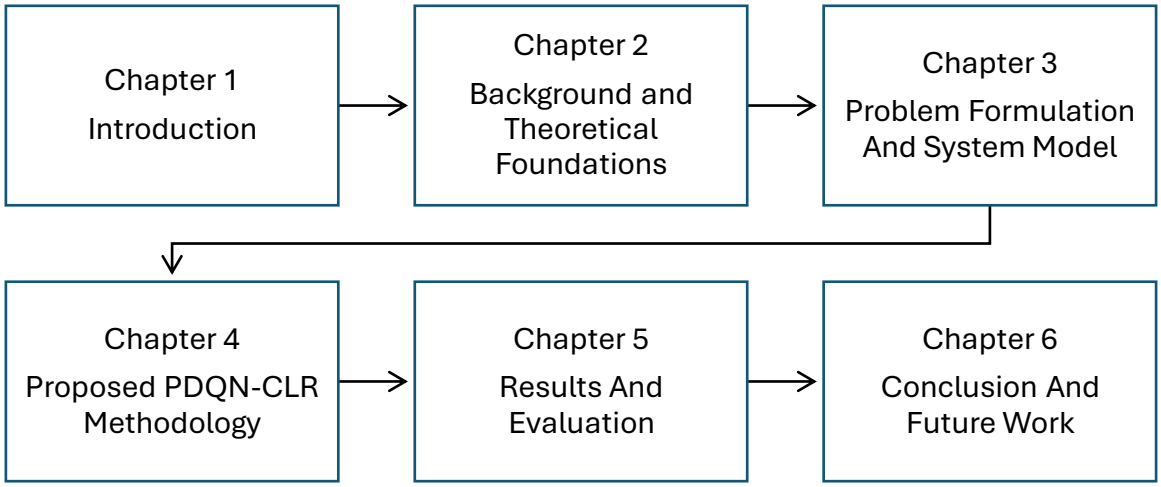

Figure 1.2: Dissertation Organization

CHAPTER TWO

BACKGROUND AND THEORETICAL FOUNDATIONS

This chapter establishes the conceptual foundations used in this dissertation. It first frames critical load restoration as a finite-horizon, step-by-step control problem for islanded distribution feeders supplied by local DERs, where dispatch and load pickup must remain feasible under the distribution power-flow constraints. It then reviews reinforcement learning through the Markov decision process framework and value-based learning, providing a basis for the Deep Q-Network methods. Next, parameterized (discrete–continuous) action spaces that match restoration decisions more directly by pairing discrete operational templates with continuous DER setpoints are introduced. Finally, it summarizes constrained RL concepts to motivate the explicit treatment of operational limits during post-disaster operations.

2.1 Critical Load Restoration Problem

Critical Load Restoration (CLR) addresses the sequential decision-making problem of restoring prioritized loads after major outages, when a distribution system is forced to operate without support from the bulk grid and must rely on local distributed energy resources (DERs). Recent resilience-oriented CLR formulations treat restoration as a multi-step control process over a finite horizon, where each control step updates DER setpoints and determines which critical loads (or fractions of loads) can be served while maintaining feasible power flow operation.

Many CLR studies adopt a simplifying but common operational assumption: that the feeder has already been reconfigured, and that re-energized (e.g., via switching and

isolation) DERs are synchronized. The subsequent challenge is the post-reconfiguration scheduling of DER dispatch and load pickup under uncertainty and resource scarcity. Table 2.1 summarizes common modeling elements used in recent DRL-based CLR studies, which motivates the problem context and challenges discussed next.

Table 2.1: Typical elements in DRL-based CLR formulations

Modeling element	What it represents in CLR	Examples in reviewed papers
Load priorities / tiers	Relative importance of loads (e.g., hospital vs residential)	Priority factors ζi for loads [22] ; multi-tier load classes (Tier I–IV) [27]
Multi-step horizon	Sequential restoration over time steps $t \in \{1, \dots, T\}$	Explicit discretized horizon and step-wise control
Decision variables	DER setpoints and load pickup/restoration levels	DER active power setpoints and power-factor angles + restored load levels
Reliability/ Monotonicity	Avoid re-shedding previously restored loads	Penalty for load shedding/restoration reversals
Network feasibility	Voltage (and sometimes thermal/power-flow) feasibility	Voltage violation penalty; feasibility handled by environment/simulator
Uncertainty	Forecast errors and uncertain renewable/DER availability	Imperfect renewable forecasts and uncertainty emphasis; uncertain DER capacity treated as unobservable state (POMDP framing) [23]

2.1.1 Post-Disaster Microgrid Operation

After extreme events, distribution feeders may be partially de-energized and segmented. When a portion of the feeder can be energized locally, the operational goal

shifts to maintaining service to critical loads using DERs within an islanded microgrid-like operating condition. In this mode, dispatchable generation, storage, and renewable DERs collectively supply available loads, and an energy management or supervisory controller determines feasible dispatch and restoration decisions across time.

Within DRL-based CLR studies, a common operational framing is: (i) an outage isolates the feeder from the main grid; (ii) the system enters a restoration window of discrete time steps; (iii) at each step, the controller selects DER dispatch and load restoration levels to maximize a resilience-aligned objective. In some formulations, the controller directly sets both DER outputs and restored load levels. In others, restoration decisions are realized through switching actions (e.g., circuit breakers), where the restoration policy can be expressed as a sequence of switch operations that energize downstream loads subject to DER availability.

Prioritization is central in post-disaster operation. In CLR formulations, critical loads are commonly assigned explicit priority factors ζi and collected in a priority vector $z = [\zeta_1, \zeta_2, \dots, \zeta_N]^T$ to encode relative importance and guide trade-offs under limited DER capacity [24]. In parallel, multi-tier categorization schemes (e.g., Tier I – Tier IV based on a load criticality index) are also used to operationalize load criticality and support systematic allocation decisions during emergencies [27].

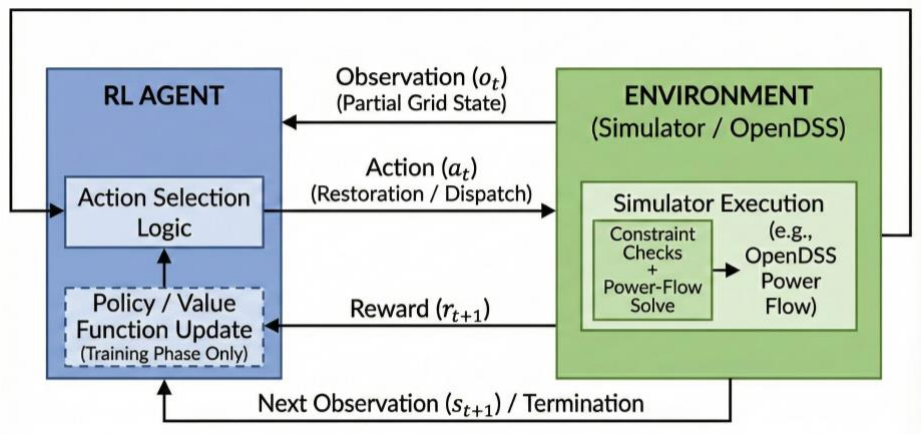

Figure 2.1: RL-driven CLR closed-loop (conceptual)

Figure 2.1 illustrates the RL-driven critical load restoration (CLR) closed-loop process: At each time step, the agent receives the grid observation (s_t), selects a restoration/dispatch action (a_t), and the simulator executes constraint checks, and a power-flow solve to generate the reward (r_{t+1}) and the next observation (s_{t+1}). This interaction pattern is explicitly adopted in CLR learning environments that follow an OpenAI Gym-style "reset/step" interface and connect the RL agent to an OpenDSS-based simulator; the environment can also enforce operational feasibility by projecting the agent's raw action into a feasible action before running the power-flow and updating states [24].

2.1.2 Challenges in Islanded Operation

CLR in islanded operation is difficult for three coupled reasons: scarcity, uncertainty, and network feasibility.

- Resource scarcity and intertemporal coupling. Islanded feeders typically have limited dispatchable capacity and storage energy. Batteries introduce state-of-charge

dynamics, and dispatchable generators can be fuel-limited, meaning decisions must consider not only instantaneous feasibility but also sustainment over the restoration horizon. This function motivates multi-step objectives that reward sustained restoration and penalize strategies that restore loads briefly and then shed them.

- Uncertainty in renewable/DER availability. Renewable DER output is uncertain, and forecasts are imperfect; this impacts the feasibility of restoration decisions and can degrade controller performance. Some CLR work treats uncertain DER generation capacity as an unobservable component of the environment state, motivating partially observable formulations, where an agent must infer the underlying condition from forecasts and history.
- Network feasibility and nonlinear power flow. Restoring load changes power flows and voltages in a nonlinear manner, particularly in unbalanced distribution feeders. For this reason, many CLR formulations incorporate voltage feasibility through a penalty term that discourages voltage violations. Studies often justify the penalty approach by noting that bus voltages are system outcomes rather than direct decision variables in a standard RL formalism.

A representative resilience-aligned objective used in CLR DRL work combines (i) priority-weighted restored load, (ii) a monotonicity/reliability term penalizing re-shedding, and (iii) a voltage violation penalty. For example, recent formulations maximize a sum across time steps of the form:

$$\max \sum_{t \in \mathcal{T}} \left(\zeta p_t - \mu - \lambda \sum_{i \in \mathcal{N}} \left(\max\left(0, v_{t,i} - v^{max}\right)^2 + \max\left(0, v^{min} - v_{t,i}\right)^2 \right) \right) \quad (1)$$

where p_t denotes restored real power at time t, v_t is the bus-voltage magnitude vector, ζi encodes load priorities, and the shedding term penalizes decreases in restored load between steps.

2.2 Reinforcement Learning Fundamentals

Reinforcement learning (RL) addresses sequential decision-making in unknown or complex environments by learning a control policy through interaction. A standard formalization is the Markov Decision Process (MDP), which captures system dynamics through a state, action, transition, and reward model [30].

2.2.1 Markov Decision Process

An MDP is commonly defined as the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where s ∈ *S* is the state, a ∈ *A* is the action, *P*(s'|s,a) is the transition model, r(s,a) is the reward, and ϒ ∈ [0,1] is the discount factor. The return from time *t* is:

$$G_t = \sum_{k=0}^{\infty} \gamma^k \, r_{t+k+1} \tag{2}$$

In CLR applications, the state typically encodes resource availability and the restoration status, while actions encode dispatch and restoration decisions. For example, [24] defines the state to include renewable forecasts, previous restoration level, storage state, remaining fuel, and a time index. The policy then outputs restoration/dispatch actions for the current step. Similarly [22] define a multi-step MDP over outage intervals, with actions that set DER power and restoration levels.

Table 2.2: Typical MDP mapping for CLR (illustrative, based on reviewed CLR papers)

MDP element	CLR interpretation	Examples in reviewed papers
State (s_t)	Forecasts + restoration history + resource states + time	Renewable forecast vector + previous fractional restoration + SOC + fuel + time
Action (a_t)	DER dispatch and/or restoration decision	DER setpoints and restored load levels; continuous breaker-operation variables for switching/restoration
Reward (r_t)	Priority-weighted restoration with feasibility/reliability terms	Priority-weighted load + shedding penalty + voltage penalty
Transition (P)	Power-flow-based evolution of grid + resource dynamics	OpenDSS-based state transition for distribution networks
Episode horizon	Restoration window (finite horizon)	Explicit multi-step restoration horizon (T)

2.2.2 Value Functions and Bellman Equation

For a policy $\pi(a \mid s)$, the state-value function and action-value function are defined as:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} |S_t = s] \quad (3)$$

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \quad (4)$$

The Bellman expectation equation expresses the recursive structure of value functions:

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s,a) \left(r(s,a,s') + \gamma V_\pi(s')\right) \quad (5)$$

and the Bellman optimality equation for the optimal value V^* is:

$$V^*(s) = max_a \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a)V^*(s')\right] \quad (6)$$

These relationships are foundational for value-based RL methods, including Q-learning and Deep Q-Networks.

2.2.3 Q-Learning

Q-learning is an off-policy method that learns the optimal action-value function $Q^*(s,a)$ by iteratively applying a temporal-difference update [31]. For a transition $(s_t, a_t, r_{t+1}, s_{t+1})$, the standard update is:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q\,(S_{t+1}, a) - Q(S_t, A_t)\right] \quad (7)$$

where α is the learning rate.

In CLR settings, Q-learning provides the conceptual route to value-based deep RL: the Q-value estimates the long-term benefit of choosing a restoration action under the current restoration state and resource conditions. Some DRL theses explicitly present the Q-update and then note that, in deep RL, Q(·) is approximated by a neural network rather than a table.

2.3 Deep Q-Networks

Deep Q-Networks (DQN) replace the tabular representation of Q (s,a) with a neural network $Q(s,a;\theta)$, enabling value-based learning in large state spaces. However, directly combining function approximation with bootstrapped targets can lead to unstable learning; DQN introduced stabilization mechanisms that became standard in deep value-based RL.

2.3.1 Function Approximation with Neural Networks

In DQN, the neural network takes a state (s) as input and outputs Q-values for available actions, enabling greedy action selection $a = arg\ max_a Q(s,a;\theta)$. The

network parameters are trained by minimizing a squared temporal-difference error over sampled transitions. In its common form, the loss uses a bootstrap target y:

$$L(\theta) = \mathbb{E}\left[\left(y - Q(s, a; \theta)\right)^2\right], \qquad y = r + \gamma \max_{a'} Q(s', a'; \theta^-). \tag{8}$$

Here θ^-denotes parameters of a separate target network (Section 2.3.2).

2.3.2 Experience Replay and Target Networks

Two mechanisms are central to stabilizing DQN training:

1. **Experience replay.** Rather than learning only from the most recent transition, DQN stores transitions in a replay buffer and samples mini-batches uniformly (or with variants) for training. This breaks temporal correlations in the data and improves sample efficiency [30]. The idea of replaying stored experiences predates modern deep RL and was formalized in early RL work as experience replay [32].
2. **Target networks.** DQN maintains a separate network with parameters θ^- to generate the bootstrap target y, updating θ^- periodically (hard update) or gradually (soft/Polyak update). This reduces harmful feedback loops between the current Q estimates and the target values.

These two components—replay buffers and target networks—also appear as core building blocks in many deep RL controllers beyond DQN (including several power-system DRL implementations), reflecting their practical importance for stable learning.

2.4 Parameterized Action Spaces

Many restoration-and-dispatch decisions in islanded microgrids are naturally hybrid, the controller must choose a discrete operational mode (e.g., which load block/zone to restore, which DER to activate), while also setting continuous parameters

(e.g., real/reactive power setpoints, droop coefficients). Parameterized action spaces formalize this structure by representing each discrete action type with its own continuous parameter vector [33].

2.4.1 Hybrid Action Space Formulation

A standard formulation defines a parameterized action as a pair:

$$a = (k, x_k), \tag{9}$$

where $k \in \{1, \dots, K\}$ is the discrete action type and $x_k \in X_k$ is the continuous parameter vector associated with type k. The overall action space is the union:

$$A = \bigcup_{k=1}^{K} \{(K, x_k) : x_k \in X_k\}, \tag{10}$$

This structure is the core idea behind "parameterized action space" deep RL [33]. In value-based learning, the Bellman optimality equation for such hybrid actions introduces a nested maximization:

$$Q(s_t, k_t, x_{k_t}) = \mathop{\mathbb{E}}_{r_t, s_{t+1}} \left[r_t + \gamma \max_{k \in \{1,\dots,K\}} \sup_{x_k \in \mathcal{X}_k} Q(s_{t+1}, k, x_k) \,\middle|\, s_t = s, a_t = (k_t, x_{k_t}) \right] \tag{11}$$

which highlights the main computational challenge: the supremum over the continuous parameters is generally intractable if done by brute force [34]. Table 2.3 summarizes how parameterized actions appear in DRL-based critical load restoration (CLR) and microgrid operation.

Table 2.3: Examples of parameterized actions in restoration-oriented microgrid control

Discrete action type k	Continuous parameters (x_k)	Example interpretation in CLR
Restore group/zone k	$x_k = [\Delta P_{DER1}, \dots, \Delta P_{DERm}]$	Choose which zone/load group to energize; tune supporting DER injections
Switch/reconfiguration k	$x_k = [P_{BESS}, Q_{BESS}]$	Execute a topology action while dispatching storage to maintain feasibility
DER mode selection k	$x_k = [P, Q]$ or droop settings	Select a DER control mode; set its operating point
Voltage support action k	$x_k = [Q_{inv}]$	Trigger Volt/VAR support and set inverter reactive power

This formulation also clarifies why naive alternatives can be inefficient: discretizing X_k can explode the number of actions, while relaxing the hybrid space into a fully continuous representation can increase complexity and weaken structure [34].

2.4.2 P-DQN Architecture

Parametrized Deep Q-Networks (P-DQN) address the $sup_{x_k \in X_k}$ difficulty by learning a deterministic mapping from states to continuous parameters for each discrete action, then evaluating discrete choices with a Q-network. In P-DQN, the action-value function is approximated as $Q(s, k, x_k; \omega)$ and a parameter (policy) network outputs $x_k(s; \theta)$ intended to approximate the maximizer of Q for each k [34].

A key idea is to represent the intractable maximizer as a learned function $x_k^Q(s)$, then approximate it with $x_k(s; \theta)$:

$$Q(s, k, x_k(s; \theta); \omega) \approx \sup_{x_k \in X_k} Q(s, k, x_k; \omega), \forall k. \quad (12)$$

P-DQN uses a DQN-like Bellman target, but replaces continuous maximization with the parameter network:

$$y_t = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \max_k Q(s_{t+n}, k, x_k(s_{t+n}; \theta_t); \omega_t), \tag{13}$$

and learns ω by minimizing a squared Bellman error.

For the parameter network, P-DQN updates θ to increase Q-values under the current ω. One commonly used objective is the negative sum across discrete actions:

$$L\Theta(\theta) = -\sum_{k=1}^{K} Q(s_t, k, x_k(s_t; \theta); \omega_t), \tag{14}$$

which encourages $x_k(s; \theta)$ to move toward parameters that make each action type look better under the critic.

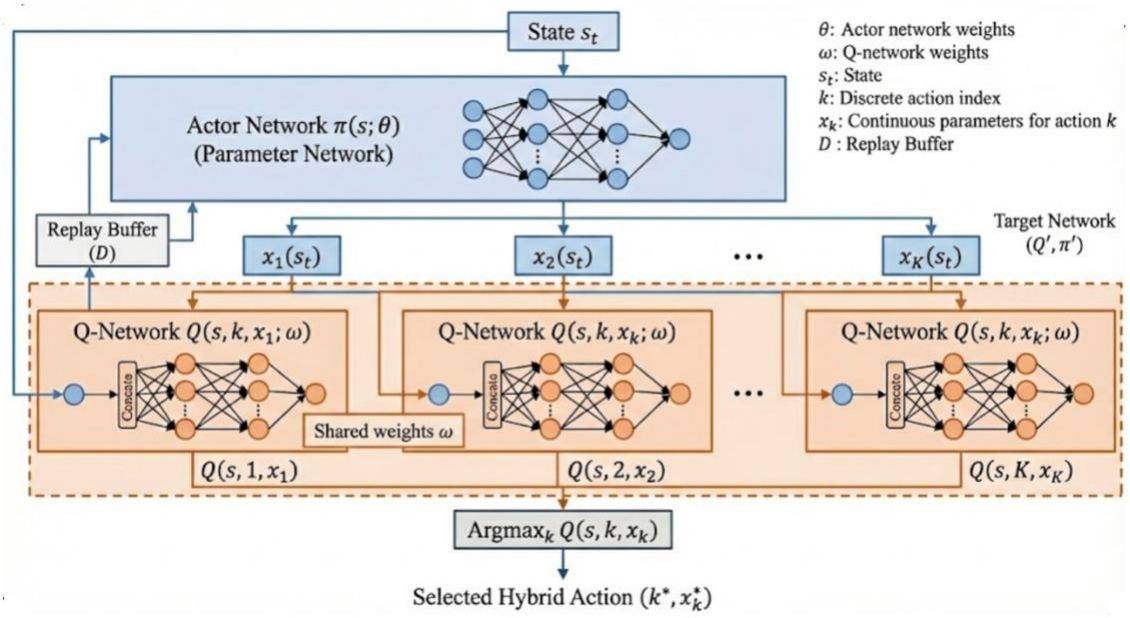

Figure 2.2: P-DQN computation graph

Figure 2.2 provides a compact view of the P-DQN dataflow that corresponds to the equations above. From the current state s_t, the parameter network generates a candidate continuous parameter vector $x_k(s_t;\theta)$ for each discrete action type $k \in \{1, \dots, K\}$, and the Q-network scores each resulting hybrid candidate $Q(s_t, k, x_k;\omega)$. The executed hybrid action is then formed by selecting $k_t = \arg\max_k Q\,(s_t, k, x_k(s_t;\theta);\omega)$ and applying its associated parameters, i.e., $a_t = \left(k_t, x_{k_t}(s_t;\theta)\right)$. During learning, transitions are stored in replay memory and the critic is trained using the DQN-style bootstrapped target y_t, while the parameter network is updated to increase the critic's value estimates via the objective $L_\theta(\theta)$ defined above [34]. For clarity, the roles of the critic, parameter network, discrete selection rule, and learning targets are summarized in Table 2.4.

Table 2.4: Core components of P-DQN

Component	Notation	Role
Q-network	$Q(s, k, x_k;\omega)$	Estimates action value for each discrete type and its continuous parameters
Parameter network	$x_k(s;\theta)$	Produces continuous parameters for each discrete action type
Discrete selection	$\arg\max_k Q\,(\cdot)$	Chooses the best action type under learned parameters
Learning targets	y_t	DQN-style bootstrapped target using $x_k(s;\theta)$
Optimization	$\mathcal{L}Q(\omega), \mathcal{L}\Theta(\theta)$	Two coupled updates (critic fit + parameter improvement)

P-DQN is explicitly designed as a value-based approach for discrete–continuous hybrid actions without approximation or relaxation, combining DQN-like discrete selection with deterministic continuous parameterization [34].

2.5 Constrained Reinforcement Learning

In post-disaster microgrid operation, feasibility constraints (e.g., voltage limits, equipment ratings, convergence/operability conditions) are not optional; they define acceptable operation. Constrained reinforcement learning (CRL) treats such requirements explicitly rather than relying only on hand-tuned penalties.

2.5.1 Constrained MDP Formulation

A Constrained Markov Decision Process (CMDP) augments the standard MDP with one or more cost signals and limits. A common single-constraint form is:

$$\max_{\pi} \; J_R^{\pi} \;\text{ s.t. }\; J_C^{\pi} \leq \alpha, \tag{15}$$

where J_R^{π} is the expected discounted return and J_C^{π} is the expected discounted (or otherwise defined) accumulated cost [35]. This framing is the foundation of modern safe/constrained RL methods and is treated systematically in CMDP theory references [36].

Table 2.5: Typical microgrid constraints as CMDP costs (examples)

Operational requirement	Example cost signal *c(s,a)*	Threshold interpretation
Voltage band $[V^{min}, V^{max}]$	magnitude of violations (e.g., sum of bus deviations beyond limits)	keep expected violation under a small bound
Thermal / power limits	overload margin beyond nameplate	limit expected overload events/severity

Operational requirement	Example cost signal *c(s,a)*	Threshold interpretation
Solver/operability constraints	non-convergence indicator or infeasibility flag	discourage trajectories that leave feasible region
Restoration policy rules	re-shedding / non-monotone restoration events	limit reversals to improve reliability

Implementation details—how costs are measured in OpenDSS and aggregated over time—fit naturally in the methodology chapter, while CMDP here provides the theoretical scaffold.

2.5.2 Lagrangian Relaxation Methods

A widely used approach solves CMDPs via Lagrangian relaxation, converting the constrained optimization into a saddle-point problem:

$$\min_{\lambda \geq 0} \max_{\theta} \; L(\lambda, \theta) = \min_{\lambda \geq 0} \max_{\theta} \; \left(J_R^{\pi\theta} - \lambda \,.\, \left(J_C^{\pi\theta} - \alpha \right) \right), \tag{16}$$

where λ is a learned penalty coefficient (Lagrange multiplier) [35]. This formulation motivates two-timescale learning: update policy parameters θ to increase the Lagrangian objective while updating λ to enforce constraint satisfaction. Reward Constrained Policy Optimization (RCPO) is a representative method that applies this Lagrangian (primal–dual) structure and explicitly discusses multi-timescale updates of θ and λ [35].

A practical reason Lagrangian methods are widely used in deep constrained RL is that they convert constraints into a learnable dual variable, enabling standard policy/value updates while adaptively enforcing constraint satisfaction [37].

2.5.3 Adaptive Multiplier Updates

Although classic Lagrangian updates are simple, they can produce oscillations and constraint overshoot during learning. PID-Lagrangian methods reinterpret multiplier

adaptation as a feedback-control problem: the cost limit is a setpoint and λ is the control input [38]. Stooke et al [38] propose updating λ using proportional–integral–derivative (PID) terms computed from the cost error. One practical discrete-time rule is summarized (Algorithm 2 in their paper) as:

- cost error: $\Delta = \mathrm{L_C} - \mathrm{d}$
- integral state: $\mathrm{I} \leftarrow \max(0, \mathrm{I} + \Delta)$
- derivative term: $\partial \leftarrow \max\left(0, \mathrm{J_C} - \mathrm{J_{C,prev}}\right)$
- multiplier: $\lambda \leftarrow max(0, K_P \Delta + K_I I + K_D \partial)$

This adds responsiveness (P-term) and anticipatory damping (D-term) while retaining the steady-state correction of the I-term; setting $\mathrm{K_P} = \mathrm{K_D} = 0$ recovers the traditional integral-only update. They also recommend stabilizing learning when λ becomes large by rescaling the Lagrangian objective into a convex combination of reward and cost gradients, helping maintain consistent parameter step sizes [38].

CHAPTER THREE

PROBLEM FORMULATION AND SYSTEM MODEL

This chapter describes the distribution network model, islanded-operation modifications, and the post-disaster resources and loads used to define the restoration problem.

3.1 IEEE 123-Bus Test System

3.1.1 Network Topology and Zone Configuration

The IEEE 123-node test feeder is a widely used benchmark for distribution-system studies and operates at a nominal voltage of 4.16 kV in a predominantly radial configuration [39]. In this study, the feeder is implemented in OpenDSS as a medium-scale network model (reported as 137 buses and 128-line segments in the adopted implementation), serving approximately 3.1 MW total load in the base configuration before post-disaster shedding. Although the benchmark is commonly referred to as a 123-node feeder, the OpenDSS realization distinguishes buses and nodes and introduces additional buses to represent device terminals and interfaces (e.g., substation connection points and multi-terminal elements, such as transformers and regulator-controlled equipment); therefore, the compiled circuit contains 137 buses in the adopted implementation.

To support structured restoration decisions, the feeder is partitioned into five zones (Zone 1–Zone 5), defined by network topology (lateral/branch groupings) and geographic locality. Each zone contains one primary grid-forming DER (Section 3.2) and a subset of critical loads (Section 3.3). Figure 3.1 presents the IEEE 123-bus feeder one-

line diagram used in this work. The five grid-forming DER buses are marked by green circles (buses 4, 26, 44, 60, and 86). The critical-load locations are distinguished by three marker styles that reflect their priority tier: P1 loads are shown as solid red squares, P2 loads as solid pink squares, and P3 loads as red hollow squares.

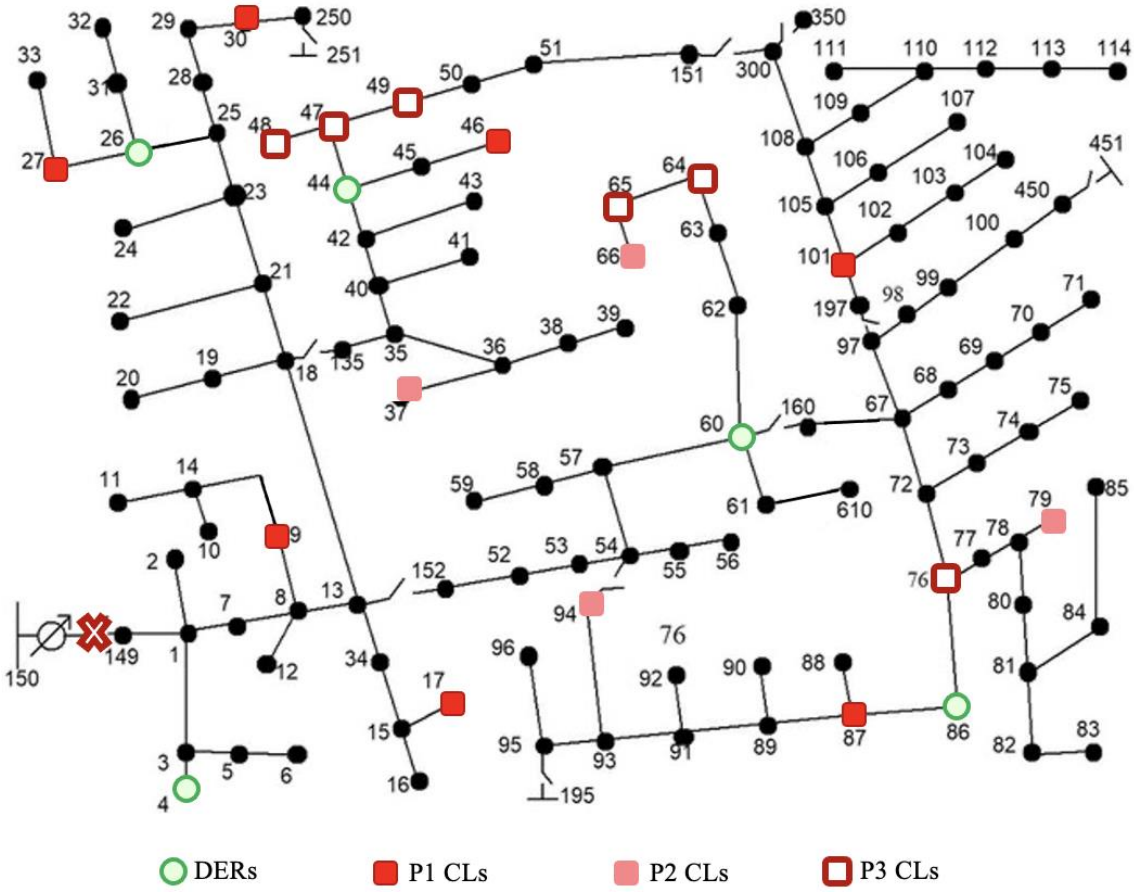

Figure 3.1: IEEE 123-node distribution feeder showing the locations of DERs and prioritized CLs (P1–P3)

3.1.2 Modifications for Islanded Microgrid Operation.

- Islanding mechanism and voltage reference.

Following a fault that isolates the feeder from the upstream grid, an islanded operating point requires a slack/voltage reference to ensure solvability of the power-flow equations. In this work, islanding is created by disabling the faulted trunk line and adding

a weak grid-forming Vsource (GFM_REF) at bus 44 with p.u. = 1.0, R1 = 0.15 Ω, and X1 = 0.6 Ω. The source impedance is intentionally selected so the reference establishes a stable voltage baseline without dominating the island, thereby preserving the sensitivity of bus voltages to DER dispatch decisions.

A simplified active-power balance for the islanded feeder can be expressed as:

$$P_{loads} \approx P_{Vsource} + \sum_{j=1}^{N_{DER}} P_{DER,j} - P_{loss}\,, \tag{17}$$

where $P_{Vsource}$ supplies the residual mismatch after DER injections and losses.

- Impedance strength and why these values were chosen.

In OpenDSS, the Vsource element is modeled as a Thevenin equivalent voltage source behind an impedance. The selected parameters (R1 = 0.15 Ω, X1 = 0.6 Ω) therefore define the reference strength seen by the islanded feeder. The corresponding Thevenin magnitude is $|Z| = \sqrt{R_1^2 + X_1^2} \approx 0.62$, which implies a three-phase short-circuit current on the order of $I_{sc} \approx V_{LL}/\left(\sqrt{3}|Z|\right) \approx 3.9kA\ at\ 4.16kV$. This level is intentionally not stiff compared with common distribution-bus fault-current magnitudes (often in the multi-kA to tens-of-kA range, depending on location and upstream strength). During environment validation, the impedance was tuned to avoid two degenerate cases: (i) near-ideal source behavior (very small |Z|), which can suppress voltage-drop sensitivity and make restoration trivially easy, and (ii) an overly weak reference (very large |Z|), which can cause widespread undervoltage and reduce solvability margin in repeated snapshot power-flow rollouts.

- Limitations of a “No Vsource” Reference Case and the Proposed Comparative Framework.

In an islanded steady-state power-flow model, a voltage reference (slack/swing) is required to define the network reference and balance losses; removing the reference makes the power-flow equations underdetermined, because the network lacks a defined voltage-angle reference and a balancing condition for real/reactive power mismatches. Accordingly, the island is modeled using an OpenDSS Vsource, which represents a Thevenin-equivalent source behind an impedance. To quantify the incremental value of DER dispatch under a fixed and well-defined reference, restoration performance is evaluated by comparing (i) a baseline case with DERs disabled (or held to non-optimizing fixed setpoints) against (ii) a controlled-dispatch case under identical faults and network conditions. This isolates the contribution of DER decisions while keeping the underlying power-flow formulation consistent across comparisons.

- DER-Attributed Critical-Load Service Criterion to Prevent Voltage-Only Over-Crediting

To ensure that restoration credit is not granted solely due to proximity to the reference bus, the environment applies a dual-criterion service evaluation for critical loads that experience a nontrivial voltage drop from Bus 44. Specifically, if the per-unit voltage drop from the Bus-44 reference to a critical-load bus exceeds 0.05 p.u., the load is counted as served only if (i) its bus voltage lies within the admissible band [0.85, 1.15] p.u. and (ii) the aggregate active-power output of nearby supporting DER(s) exceeds a small minimum threshold (10 kW) to attribute service to actual DER support rather than passive voltage hold by the reference. For loads with voltage drop ≤ 0.05 p.u., voltage

admissibility alone determines service. The 0.85–1.15 p.u. operating band is consistent with ranges commonly used in islanded operation and restoration contexts. The 5% drop trigger aligns with widely used engineering guidance that treats ~5% as a practical threshold for non-negligible steady-state voltage drop along feeders/branches.

- Empirical check that DER dispatch changes outcomes under the chosen reference strength.

With this scoring rule in place, the island reference does not automatically "serve" all loads: loads electrically close to Bus 44 can meet voltage limits without substantial DER support, but loads farther away require active DER injection to satisfy both the voltage condition and the DER-attribution threshold. In the code's fault-scenario sanity checks, disabling optimized DER dispatch reduces the achievable restoration score by roughly 9–18 percentage points (baseline vs. with-optimization), confirming that dispatch decisions are not masked by the reference model.

- Post-disaster load shedding for problem focus.

To isolate the critical-load restoration objective, non-critical loads are physically disconnected at episode start, reducing the network from 87 loads (3,134 kW) to 17 critical loads (1,088 kW) that remain connected throughout the episode. This ensures the RL agent's actions primarily influence critical-load service quality through DER injections and resulting voltage profiles (rather than bulk load composition changes).

3.1.3 Simulation Mode Selection and Computational Tradeoffs

OpenDSS supports several solution modes; in this study, the RL environment evaluates each decision step using Snapshot power-flow solutions (Mode=0) rather than time-domain Dynamics simulation (Mode=6) [40]. Snapshot mode performs one steady-

state circuit solution per `Solve` call and does not automatically advance time or sample time-series monitors unless scripted externally. This steady-state evaluation matches the learning requirement of executing large numbers of environment steps and is consistent with the modeling assumption that fast electrical transients settle between coarse dispatch intervals (30 minutes in the adopted configuration).

Because steady-state islanded power flow requires a voltage reference and loss-balancing source, the island is modeled with a Thevenin-equivalent Vsource (slack/reference) behind impedance, as described in the OpenDSS element definition and developer guidance. Under the fixed reference described in Section 3.1.2, Table 3.1 reports a baseline-versus-dispatch check showing that restoration outcomes change materially with DER setpoints, supporting that the learned policy contributes beyond passive reference support.

Dynamics mode provides higher physical fidelity because it integrates device state variables over time and performs circuit solutions at each integration step. In OpenDSS, each dynamics time step executes a predictor–corrector sequence that calls `SolveSnap` twice per step. The time step size is user-defined (StepSize), and typical dynamics studies use sub-millisecond steps (e.g., ~0.2–1.0 ms) to resolve fast transients and control behavior. At these step sizes, Dynamics mode requires on the order of 2,000–10,000 circuit solves per simulated second (two solves per time step).

For the 30-minute decision interval and 72-step horizon used here, Snapshot requires 72 circuit solves per episode, whereas Dynamics would require approximately 259 million to 1.30 billion circuit solves per episode (depending on the chosen dynamic step size), and correspondingly ~7.8×10^{11} to 3.9×10^{12} solves over 3,000 training episodes.

Using the measured per-step Snapshot solve time reported in Chapter Four (≈5 ms per solve), the implied training cost under Dynamics would increase from hours to an infeasible multi-decade scale, even before accounting for additional iterations and device-model overheads in dynamics simulation.

Accordingly, the reported results establish feasibility and restoration behavior at steady-state operating points (voltage profiles, power balance, and constraint satisfaction) rather than transient stability, frequency dynamics, or sub-second protection interactions. These time-domain phenomena are outside the scope of the present steady-state RL training study and motivate future work that validates the learned restoration policies under Dynamics-based simulation as a complementary assessment.

Table 3.1: Baseline-versus-dispatch restoration check under the same Vsource reference

Item	Definition (same island reference)	Typical observed effect in fault checks
Baseline (no optimization)	Vsource present; DERs disabled or held at a fixed non-optimizing setting	Lower restoration score
Optimized dispatch	Vsource present; DER setpoints controlled by the agent	Higher restoration score
Dispatch impact	Difference between the two cases above	~0.09 to ~0.18 absolute score improvement (scenario-dependent)
Anti-masking safeguard	Service credit requires DER attribution when $\Delta V(\text{ref}\rightarrow\text{bus}) > 0.05$ p.u.	Prevents voltage-only over-crediting on remote buses

3.2 Distributed Energy Resources (DERs) Configuration

3.2.1 DER Specifications and Placement

Five battery energy storage systems (BESS) are placed to provide geographically distributed support across five zones. The rated active/reactive power and energy capacities are listed in Table 3.1.

Table 3.2: DER specifications and placement

DER	Bus	Zone	P_{max} (kW)	Q_{max} (kVar)	Energy (kWh)
DER1	4	Southwest (Zone 1)	125.33	64.25	780.65
DER2	26	Northwest (Zone 2)	196.23	86.52	539.78
DER3	44	Northeast (Zone 3)	129.52	89.02	919.12
DER4	60	Central (Zone 4)	92.63	54.56	895.29
DER5	86	South (Zone 5)	101.63	152.89	550.23
Total	—	—	**645.34**	**447.24**	**3,685.07**

3.2.2 DER Power Injection Model

- OpenDSS component model

Each DER is implemented as an OpenDSS Storage element with parameters defining inverter rating and battery capacity. The implementation uses kWRated, kWhRated, and initial stored energy kWhStored based on initial State of Charge (SOC), with charge/discharge efficiencies set to 95%. The Storage element supports dispatch behaviors including “FOLLOW” mode; OpenDSS documentation describes how output follows the active multipliers in this mode and how charging/discharging relates to the rated kW [40].

- Post-Disaster Capacity Derating

Post-disaster resource availability is modeled using a capacity factor α in$[0,1]$

applied to each DER's rated capacity:

$$P_{j,avail} = \alpha\, P_{j,max}, \tag{18}$$

where $P_{j,avail}$ is the usable post-disaster active-power capability for DER (j).

• Dispatch Scaling from Normalized Actions to Physical Setpoints.

The continuous action parameters are produced in normalized form and scaled to physical setpoints through a bounded mapping with floor/ceiling fractions. For active power, the model applies a minimum dispatch fraction "floor" of 0.10 and a maximum fraction "ceiling" of 0.80 (both configurable), then multiplies by derated capacity:

$$P_{j,t} = \left(f_{min} + u_{j,t}^{P}(f_{max} - f_{min})\right) \alpha P_{j,max}, u_{j,t}^{P} \in [0,1], \tag{19}$$

$With\ f_{\min} = 0.10, f_{\max} = 0.80$. Reactive power is scaled to a symmetric range:

$$Q_{j,t} = \left(2u_{j,t}^{Q} - 1\right) Q_{j,max}, \quad u_{j,t}^{Q} \in [0,1]. \tag{20}$$

This injection model directly drives the OpenDSS power-flow engine at each step via updated DER setpoints.

3.2.3 Operational Constraints on Capacity and Ramp Rate

DER dispatch is constrained by inverter ratings, energy availability (SOC), and ramping limits. In this study, constraints are enforced in two layers: (i) proactive clipping/blocks before executing OpenDSS, and (ii) post-solution penalties for network constraints, such as voltage limits.

Table 3.3: Operating constraints for islanded DER dispatch

Category	**Constraint**
Capacity (active)	$0 \le P_{j,t} \le \alpha P_{j,max}$
Capacity (reactive)	$-Q_{j,max} \le Q_{j,t} \le Q_{j,max}$
SOC reserve	Dispatch is blocked if $SOC_{j,t} < 0.10$

Category	Constraint
Ramp rate	$\left\|P_{j,t} - P_{j,t-1}\right\| \leq \rho, P_{j,max}, \text{with} \rho = 0.05 \text{per step}$
Voltage (network)	$0.85 \leq V_i \leq 1.15$ p.u. (observed after power flow)

The OpenDSS snapshot solver is called after applying the DER-side feasibility checks [40]. Network voltage constraints are evaluated from the solved bus voltages.

The voltage bounds are relaxed to 0.85–1.15 p.u. (±15%), rather than the ANSI C84.1 Range A service-voltage band of 0.95–1.05 p.u. (±5%), to represent emergency islanded restoration conditions in which sustaining critical loads is prioritized and tight regulation is constrained by the lack of a stiff grid reference and limited DER capacity [41].

3.3 Critical Load Classification

3.3.1 Priority Tier Definition

Critical loads are modeled as a fixed set of 17 loads categorized into three priority tiers, each assigned a weight ζ that represents its relative importance in restoration objectives. Table 3.3 summarizes the tiers and their aggregate demand.

Table 3.4: Three-tier critical load structure

Tier	Weight ζ	Count	Total P (kW)	Description
P1, Critical	1.0	7	246.27	Hospitals, emergency services
P2, Essential	0.7	4	67.13	Essential public services
P3, Important	0.4	6	775.00	Large commercial loads

The full list of critical loads $(\text{bus}, \text{zone}, (P/Q))$ is defined explicitly in the system configuration and remains connected throughout the episode (after non-critical shedding).

3.3.2 Load-to-DER Zone Mapping

Each critical load is mapped to a zone based on feeder topology and proximity to the zone's primary DER. Most loads are assigned to exactly one zone; a special case is included for a multi-zone critical load (bus 101), reflecting shared support expectations between Zones 4 and 5.

Table 3.5: Zone-level mapping between DERs and critical loads (IDs)

Zone	Primary DER (Bus)	P1 loads	P2 loads	P3 loads
Zone 1 (Southwest)	DER1 (Bus 4)	CL-1 (Bus 9), CL-2 (Bus 17)	—	—
Zone 2 (Northwest)	DER2 (Bus 26)	CL-3 (Bus 27), CL-4 (Bus 30)	CL-8 (Bus 37)	—
Zone 3 (Northeast)	DER3 (Bus 44)	CL-5 (Bus 46)	—	CL-12 (Bus 47), CL-13 (Bus 48), CL-14 (Bus 49)
Zone 4 (Central)	DER5 (Bus 60)	CL-7 (Bus 101)*	CL-9 (Bus 66)	CL-15 (Bus 64), CL-16 (Bus 65), CL-17 (Bus 76)
Zone 5 (South)	DER4 (Bus 86)	CL-6 (Bus 87), CL-7 (Bus 101)*	CL-10 (Bus 79), CL-11 (Bus 94)	—

*CL-7 is treated as multi-zone (Zones 4 and 5) in the configuration.

3.4 Fault Scenario Generation

3.4.1 Fault Location and Severity Model

Post-disaster disturbances are represented using a small set of trunk-line faults located near the substation, selected to produce severe operating conditions and large-scale loss of upstream support. In the implemented environment, three faulted-line

candidates are considered: Sw1, L115, and L3. Each fault is characterized by its affected line segment connecting two buses, the network partitioning it induces into upstream and downstream regions, and a qualitative severity label of Extreme, reflecting that these main-trunk interruptions occur close to the source and enforce complete islanding behavior.

Table 3.6: Trunk-line fault scenarios used for islanding analysis

Fault Line	Location	Buses Upstream	Buses Downstream	Severity
Sw1	150r → 149	150 (substation)	All 123 buses	Extreme
L115	149 → 1	150, 149	All except 149	Extreme
L3	1 → 7	150, 149, 1, 2, 3, 4, 5, 6	Most loads	Extreme

This fault set is intentionally concentrated on trunk segments near the substation, providing a consistent and challenging restoration setting in which the controller must rely on distributed resources rather than on upstream grid support.

3.4.2 Post-Fault Island Formation

For each episode, islanding is produced by disabling the selected faulted line element in OpenDSS, leaving the rest of the feeder intact. This aligns with the standard OpenDSS workflow in which an element can be removed from the circuit solution while remaining defined in memory (i.e., it can be re-enabled later if desired) [40].

Let $e_f \in \mathcal{E}$ denote the faulted line selected from the candidate set {Sw1, L115, L3}. Island formation is implemented as:

$$\mathcal{E}^{post} = \mathcal{E} \setminus e_f, \tag{21}$$

followed by solving a steady-state power flow under the island reference configuration. Operationally, the sequence is:

1. Disable the faulted line (remove e_f from the active circuit) [40].
2. Establish a grid-forming voltage reference using the weak Vsource at bus 44 (Section 3.1.2), then solve the power flow.

This procedure provides a consistent post-fault topology for learning and evaluation while ensuring that dispatch actions from the DERs remain the dominant controllable mechanism for improving voltages and restoring service quality across the islanded feeder.

3.5 Resource Availability Scenarios

To capture uncertainty in post-disaster resource conditions, DER availability is modeled via a capacity factor α, defined as the fraction of nameplate active-power capability that remains usable after the event. The effective available active power for DER (j) is:

$$P_j^{avail} = \alpha\, P_j^{rated}, \qquad \alpha \in [0,1]. \tag{22}$$

Two scenario families are used: a sufficient regime in which the total available DER power slightly exceeds the minimum support required for full restoration, and a scarce regime in which the available DER power is far below the required level.

3.5.1 Sufficient Resource Scenario

The primary evaluation regime is the Sufficient scenario with $\alpha \in [0.45,0.55]$. In this setting, aggregate DER availability is reported as 290–355 kW against an estimated 285 kW required to support full restoration, corresponding to 102–125% of the needed support. Because the available capacity is comparable to the needed level, full restoration is feasible in principle, and learning emphasizes optimization (e.g., efficient dispatch and voltage-feasible setpoints).

3.5.2 Scarce Resource Scenario

The Scarce scenario uses $\alpha \in [0.11, 0.14]$, producing aggregate DER availability of 71–90 kW. Relative to the same 285 kW requirement, this corresponds to 25–32% of needed capacity, and therefore full restoration is not achievable. Under scarcity, learning is driven by prioritization—allocating limited DER output to maximize service of higher-priority loads over the episode horizon.

Table 3.7: Resource availability regimes used in experiments

Scenario	α	Available DER	Required	Availability	Full restoration
Sufficient	0.45–0.55	290–355 kW	285 kW	102–125%	Yes (possible)
Scarce	0.11–0.14	71–90 kW	285 kW	25–32%	No (infeasible)

3.5.3 Randomized Capacity Degradation Model

To avoid overfitting to a single deterministic setting, each training episode samples uncertainty in three components:

- Fault scenario (F) sampled from $\{L115, Sw1, L3\}$ with weights [0.4, 0.4, 0.2].
- Capacity factor α sampled uniformly within the scenario's bound (Scarce or Sufficient), representing uncertain post-disaster derating.
- Initial state-of-charge $SOC_{j,0}$ sampled in the range 0.8–1.0 for each DER, capturing uncertainty in pre-event usage.

The episode length is configured as a 36-hour restoration window with 72 steps and a 30-minute timestep, so SOC depletion becomes a binding operational driver rather than a negligible constraint.

3.6 Notation Summary

Table 3.8: Symbols used in Chapter Three

Symbol	Meaning
$\mathcal{N}, \mathcal{E}$	Sets of buses and line elements in the feeder
$t \in \{0, \dots, T-1\}$	Decision timestep index, with (T=72)
Δt	Timestep duration (30 minutes)
$j \in \{1, \dots, 5\}$	DER index (five BESS units)
P_j^{rated}, Q_j^{rated}	DER rated active/reactive power limits
α	Capacity factor (post-disaster derating fraction)
P_j^{avail}	Available DER active power after derating, $P_j^{\text{avail}} = \alpha\, P_j^{\text{rated}}$
$SOC_{j,t}$	State-of-charge of DER (j) at time (t)
e_f	Faulted line element selected for islanding
F	Random fault scenario variable (L115,Sw1,L3)

3.7 Chapter Summary

This chapter defined the system model used throughout the dissertation. The IEEE-123 feeder is converted to an islanded microgrid by applying trunk-line fault scenarios and disabling the selected line element in OpenDSS, while maintaining solvability via a grid-forming reference configuration. Five grid-forming BESS DERs and 17 tiered critical loads form the restoration setting, and post-disaster uncertainty is represented through two resource regimes (Sufficient and Scarce) governed by a capacity-factor derating model. Episode-wise randomization of fault type, capacity factor, and initial SOC—over a 36-hour horizon—creates a challenging decision problem in which dispatch must balance immediate restoration against energy depletion and feasibility limits.

CHAPTER FOUR

PROPOSED PDQN-CLR METHODOLOGY

This chapter presents PDQN-CLR, the proposed deep reinforcement learning controller for post-disaster critical-load restoration in the islanded feeder. It first motivates why restoration control is inherently hybrid, requiring discrete operational choices together with continuous DER real/reactive setpoints, and therefore benefits from a parameterized action formulation. The chapter then maps the restoration task to an MDP and details the PDQN learning stack (coupled Q- and parameter networks) used to score discrete candidates while generating the continuous setpoints. Next, it specifies the full PDQN-CLR architecture, including constraint handling via adaptive penalties and a safe-execution layer that projects actions to maintain feasibility before OpenDSS evaluation. Finally, it documents the training methodology and settings (replay, target tracking, exploration schedule, and hyperparameters) used in the experiments.

4.1 Motivation for Hybrid Action Space in Power System Control

Post-disaster critical load restoration is a control problem with mixed decision modalities. At each decision epoch, the operator must (i) select what operational move to apply (e.g., which local support mode or dispatch focus to trigger) and (ii) decide how much active and reactive power each grid-forming resource should inject to correct voltage drops and energize priority loads. This structure is neither purely discrete nor purely continuous; it is inherently hybrid.

Hybrid control is widely recognized in distribution operation tasks where legacy devices introduce discrete decisions (e.g., regulator tap positions, capacitor switching)

while inverter-based resources provide continuous setpoints (e.g., reactive power support). Recent learning-based Volt–VAR formulations explicitly model this as a hybrid action space with discrete device operations and continuous inverter control, reinforcing that a hybrid abstraction is natural for distribution automation [42].

In this study, the restoration agent is designed to operate the islanded IEEE 123-bus feeder by coordinating discrete restoration choices with continuous DER setpoints, consistent with the physical decision pipeline implemented in the project environment.

4.1.1 Limitations of Existing RL Approaches

A practical restoration controller must represent the action as a coupled pair:

$$a_t = (k_t, X_{K_t}, t), \tag{23}$$

where k_t is a discrete choice and X_{K_t}, t is a continuous parameter vector. When discrete-only or continuous-only RL methods are applied to this hybrid decision problem by discretizing continuous setpoints or relaxing discrete choices, the resulting controller typically suffers from either combinatorial growth in the effective action space or non-smooth optimization artifacts. Parameterized action formulations address this by modeling the action explicitly as a coupled pair (k_t, X_K) [33].

Table 4.1: Limitations of common RL baselines for restoration control

Baseline family	Core strength	Restoration Mismatch	Typical consequence
Value-based discrete control (DQN)	Strong discrete decision-making	Cannot natively output continuous setpoints	Requires discretizing continuous injections, causing combinatorial growth in the effective action count [6]

Baseline family	Core strength	Restoration Mismatch	Typical consequence
Deterministic continuous control (DDPG)	Continuous setpoints	Discrete operational choices must be forced into R and rounded	Non-smooth learning signal and brittle behavior around discrete boundaries [43]
On-policy policy gradient (PPO)	Stable updates and broad applicability	Hybrid action parameterization must be engineered carefully; sample cost can become high for simulator-in-the-loop training	Reduced data efficiency in long-horizon restoration episodes [44]
Maximum-entropy continuous control (SAC)	Strong continuous-control baseline	Same discrete-action incompatibility as DDPG unless reformulated	Requires relaxation or auxiliary discrete handling [45]

These issues are not abstract: in the implemented IEEE-123 environment, the discrete choice set contains 21 options while the continuous control layer requires 10 setpoints per step (active and reactive power for five DERs). Discretizing continuous DER setpoints converts a low-dimensional continuous control into a combinatorial discrete action space whose size grows exponentially with the number of controlled parameters, which quickly becomes impractical for multi-DER dispatch.

4.1.2 Parameterized Action Space Formulation

To represent hybrid restoration decisions without discretization artifacts or continuous relaxations, this study adopts the parameterized action space model. Following the canonical formulation, the action space is defined as a union of continuous manifolds indexed by discrete actions:

$$\mathcal{A} = \bigcup_{k \in \mathcal{K}} k \times \mathcal{X}_{\mathrm{k}}, \tag{24}$$

where $\mathcal{K}$ is the discrete action set and $\mathcal{X}_{\mathrm{k}} \subset \mathrm{R}^{\mathrm{d_k}}$ is the continuous parameter domain associated with action k [33].

This structure is the methodological foundation behind P-DQN, which learns (i) how to choose k and (ii) how to generate $\mathrm{x_k}$ conditioned on both the state and the candidate action [34]. In PDQN-CLR, the hybrid decision is aligned with power-system semantics: the discrete layer selects an operational template (restoration mode), while the continuous layer specifies DER injections $\mathrm{x_k} = [\mathrm{P_1}, \mathrm{Q_1}, \ldots, \mathrm{P_5}, \mathrm{Q_5}]^{\mathrm{T}}$ that implement the chosen template in physical units after scaling.

4.2 MDP Formulation for Critical Load Restoration

The post-disaster restoration problem is modeled as a Markov Decision Process (MDP) with tuple:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathrm{r}, \gamma), \tag{25}$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the parameterized action space, $\mathcal{P}(\mathrm{s}'|\mathrm{s}, \mathrm{a})$ are the transition dynamics induced by the OpenDSS power-flow simulator and restoration logic, *r* is the shaped restoration reward, and $\gamma \in (0,1)$ is a discount factor. Standard MDP definitions follow established RL references [30].

4.2.1 State Space Design

The state vector is constructed to capture (i) network electrical conditions, (ii) DER operating points and limits, and (iii) restoration progress for the 17 critical loads. The implementation uses a 209-dimensional state:

$$s_t \in \mathbb{R}^{209}. \tag{26}$$

Table 4.2 summarizes the state components, and the normalization applied to each feature group.

Table 4.2: State vector composition

Component group	Dimension	Summary
Bus voltage magnitudes	125	Per-unit voltages for IEEE-123 buses used by the environment
DER operating point	10	Current (P/Q) outputs for five DERs
DER capability limits	10	Available (P/Q) bounds including SOC-aware active power limit
Critical-load status and demand	51	Served flags (17) plus (P/Q) demands (17 + 17)
Zone and context features	8	Zone voltage summaries and episode context indicators
SOC features	5	Storage state of charge per DER
Total	209	---

This design supports learning under resource scarcity because it exposes both instantaneous feasibility (voltages, power limits) and intertemporal constraints (SOC).

4.2.2 Hybrid Action Space Design

The action at time (t) is parameterized as $a_t = (k_t, x_{k_t,t})$ with:

- Discrete choice: $k_t \in \{0, \dots, 20\}$ (21 discrete action templates).
- Continuous parameters: $X_{k_t,t} \in \mathbb{R}^{10}$, representing $[P_1, Q_1, \dots, P_5, Q_5]^T$. The parameter ordering (1…5) follows the fixed DER ordering by bus location {4,26,44,60,86}, consistent with the DER placement and ratings tables. The decision is modeled as a two-stage hybrid action: selecting a discrete action index $k_t \in \{0, \dots, 20\}$ followed by assigning continuous DER setpoints $X_{k_t,t} \in \mathbb{R}^{10}$ for five DER units. This structure is

summarized in Table 4.3 and aligns with the parameterized action space formulation used in P-DQN. Continuous parameters are produced in normalized form and scaled to physical setpoints using the available post-disaster capacity factor α and DER ratings:

$$P_i^{\text{cmd}} = \hat{P}_i \; \alpha \; P_i^{\max}, \tag{27}$$

consistent with the scaling procedure used in the project environment.

Table 4.3: Discrete action categories in the PDQN-CLR hybrid action space

ID range	Category	Count	Description
0–4	Single DER	5	Activate or adjust an individual DER
5–9	Zone dispatch	5	Coordinate zone-level dispatch
10–11	Priority restore	2	Target priority load groups
12–17	Multi-zone	6	Coordinate pairs of zones
18–19	Voltage support	2	Voltage regulation actions
20	No-op	1	Maintain current dispatch

The complete action-template definitions and involved-DER sets are provided in Appendix A (Table A.1). Although templates 0–4 and 5–9 can map to the same involved-DER mask, they are retained as distinct discrete labels (device-centric vs. zone-centric) so the value function can learn context-dependent preferences conditioned on the discrete action identity. At each decision step, the continuous parameter vector is applied only to the DER(s) targeted by the selected discrete action k_t; all non-targeted DERs retain their previous setpoints (P,Q) from step (t-1).

After Table 4.3 categorizes the discrete decision layer, the second component of the hybrid action specifies the continuous dispatch levels. Once the agent selects a discrete action index k_t, it determines the associated continuous parameter vector that maps this high-level choice into physical power injections. In PDQN-CLR, the

continuous parameters correspond to active and reactive power commands for the five grid-forming DERs:

$$\mathrm{x}_{k_t,t} = \left[P_{1,\mathrm{t}}, \mathrm{Q}_{1,\mathrm{t}}, \mathrm{P}_{2,\mathrm{t}}, \mathrm{Q}_{2,\mathrm{t}}, \mathrm{P}_{3,\mathrm{t}}, \mathrm{Q}_{3,\mathrm{t}}, \mathrm{P}_{4,\mathrm{t}}, \mathrm{Q}_{4,\mathrm{t}}, \mathrm{P}_{5,\mathrm{t}}, \mathrm{Q}_{5,\mathrm{t}}\right]^{\mathsf{T}} \in \mathbb{R}^{10}. \quad (28)$$

Each component is produced in normalized form and scaled to feasible physical setpoints using DER ratings and the post-disaster capacity factor α.

4.2.2.1 Action-Template Decoding and Selective Dispatch Update

Each discrete template *k* is interpreted as a targeting mask over the DER fleet. Let $D = 1, \ldots, 5$ denote the DER index set and $I(k) \subseteq D$ the subset involved by template *k*. At decision step *t*, the continuous parameter vector provides candidate setpoints for all five DERs, but only DERs in $I(k_t)$ are overwritten; all others retain their previous operating points $\left(P_{j,t-1}, Q_{j,t-1}\right)$. This selective-update rule avoids unintended setpoint resets when the action targets a single device or a limited subset.

Step A: Normalized-to-physical scaling. The parameter network outputs normalized components in [0,1], which are mapped to feasible physical commands using DER ratings and the post-disaster capacity factor \alpha. Active power is mapped through the bounded floor/ceiling schedule used in the DER injection model, and reactive power is mapped symmetrically to $\left[-\mathrm{Q}_{\mathrm{max,j}}, +\mathrm{Q}_{\mathrm{max,j}}\right]$.

Step B: Template involvement filtering. Let $\left(P_{j,t,}^{cand} Q_{j,t}^{cand}\right)$ denote the candidate setpoints after scaling for DERj. The executed dispatch is:

$$\left(P_{j,t},\, Q_{j,t}\right) = \begin{cases} \left(P_{j,t}^{cand},\, Q_{j,t}^{cand}\right), & j \in I(k_t) \\ \left(P_{j,t-1},\, Q_{j,t-1}\right), & j \notin I(k_t) \end{cases}$$

Thus, templates differ operationally by the subset *I(k)* that is permitted to change at that step.

Step C: Safe execution and constraint enforcement. Before OpenDSS execution, DER-side feasibility checks enforce capacity, SOC, and ramp constraints; remaining network violations are assessed after the power-flow solution. OpenDSS then evaluates the steady-state operating point under the updated injections

4.2.3 Priority-Weighted Reward Structure

Prioritized critical-load restoration is commonly formulated as maximizing a priority-weighted amount of served demand over a finite restoration horizon. Following this convention, the general objective can be written as:

$$\max_{\{\mathbf{p_t}\}_{\mathbf{t=1}}^{\mathbf{T}}} \sum_{t=1}^{T} \mathbf{w}^{\top} \mathbf{p_t} \tau, \tag{29}$$

where w denotes load priority weights, p_t represents the vector of credited served demand at step *t*, and τ is the control interval [46]. In this work, restoration is not modeled as critical-load pickup through switching. Instead, the environment applies a two-stage load-management design: non-critical loads are physically disconnected at episode initialization, while the 17 critical loads remain electrically connected throughout the episode and are evaluated at every step using a service-quality test derived from the post-dispatch OpenDSS power-flow outcome.

To avoid awarding service credit solely due to proximity to the island reference, the service indicator used in the reward combines voltage admissibility with DER-support attribution. For a critical-load bus whose per-unit voltage drop relative to the island reference bus (Bus 44) exceeds 0.05 p.u., service credit is granted only when (i)

the bus voltage lies within the admissible operating band and (ii) the supporting zone DER output exceeds a minimum threshold (10 kW). For buses within 0.05 p.u. of the reference, service credit follows voltage admissibility alone. This prevents electrically distant buses from being over-credited by the reference source and ensures that improvements in restoration score reflect DER dispatch decisions.

Accordingly, each critical load *i* is represented by a binary service indicator $s_i(t) \in \{0,1\}$, where $s_i(t) = 1$ if load *i* is classified as served under the hybrid service evaluation used in this —i.e., the bus voltage satisfies the admissible operating limits, and (for electrically distant critical buses) service credit is granted only when the voltage support is attributable to active DER dispatch rather than slack-source masking—otherwise $s_i(t) = 0$. The credited served demand is expressed as $p_{i,t} = s_i(t), p_i$, where p_i is the fixed demand of critical load *i*. To quantify restoration quality in a normalized, priority-aware form, the Priority-Weighted Critical Score (PCS) is defined as:

$$\mathrm{PCS}(t) = \frac{\sum_{i \in \mathcal{C}} \zeta_i \; s_i(t)}{\sum_{i \in \mathcal{C}} \zeta_i}, \tag{30}$$

where $\mathcal{C}$ is the set of critical loads and ζ_i is the priority tier weight. The step reward is then shaped around $\mathrm{PCS}(t)$ to emphasize priority-weighted service recovery, while discouraging infeasible operating points through constraint-related penalties. In particular, DER-side feasibility is enforced proactively via the safe execution layer (capacity, SOC reserve, and ramp-rate limits), whereas voltage violations—observable only after power-flow execution—are penalized through the adaptive Lagrangian constraint handler described in Section 4.4.2.

4.2.4 Step-Reward Formulation with Constraint Penalties

To convert the restoration objective in Section 4.2.3 into a learning signal, PDQN-CLR defines a scalar step reward that prioritizes (i) priority-weighted critical load service; (ii) active DER contribution (to reduce sparsity early in learning); (iii) Priority-1 restoration preference; and (iv) numerical feasibility (power-flow convergence). Constraint satisfaction is encouraged by subtracting adaptive penalty terms for voltage, capacity, and ramp violations, consistent with the constraint-handling module described later in Section 4.4.2.

At time step (t), the reward is defined as:

$$r_t = r_t^{base} - r_t^{pen}, \tag{31}$$

where the base reward is:

$$r_t^{base} = w_{pcs}\, PCS(t) + w_{dispatch}\, D(t) + w_{priority}\rho_{P1}(t) + b_{conv}(t), \tag{32}$$

and the penalty term is:

$$r_t^{pen} = \lambda_v(t)\, V_{viol}(t) + \lambda_c(t)\big(C_{viol}(t)\big)^2 + \lambda_r(t)\, R_{viol}(t). \tag{33}$$

This reward form (base components and penalty structure) follows the project implementation.

Coefficient settings.

The reward function is parameterized using the coefficient vector $\{w_{\text{pcs}}, w_{\text{dispatch}}, w_{\text{priority}}\} = \{50, 20, 10\}$ which is adopted consistently across all experiments reported in this chapter. The convergence bonus is:

$$b_{\text{conv}}(t) = \begin{cases} +1, & \text{if the OpenDSS power flow converges,} \\ -5, & \text{otherwise.} \end{cases} \tag{34}$$

Component definitions.

- PCS(t) is the normalized priority-weighted restoration score defined in Section 4.2.3.
- *D(t)* is the dispatch contribution ratio:

$$\mathrm{D(t)} = \frac{\text{total DER active-power output at t}}{\text{total critical-load active-power demand}}, \tag{35}$$

providing reward signal even before full restoration is achieved.

- $\rho_{\mathrm{P1}}(\mathrm{t})$ is the served fraction of Priority-1 critical loads:

$$\rho_{\mathrm{P1}}(\mathrm{t}) = \frac{\mathrm{N_{P1\ served}(t)}}{\mathrm{N_{P1\ total}}}. \tag{36}$$

Violation measures.

The three violation signals are computed as follows (as implemented in the project environment):

- $\mathrm{V_{viol}(t)}$: aggregate voltage deviation outside limits 0.85 – 1.15 p.u.
- $\mathrm{C_{viol}(t)}$: aggregate DER capacity exceedance in kW, penalized quadratically.
- $\mathrm{R_{viol}(t)}$: aggregate ramp-rate exceedance in kW/step.

Adaptive multipliers.

$\lambda_{\mathrm{v}}(\mathrm{t}), \lambda_{\mathrm{c}}(\mathrm{t}), \lambda_{\mathrm{r}}(\mathrm{t})$ are adaptive Lagrange multipliers updated from violation history (bounded within a preset range), so the agent can explore early while repeated infeasibilities are increasingly discouraged.

4.3 PDQN Training Framework

4.3.1 Dual-Network Architecture Overview

PDQN-CLR employs two coupled function approximators:

- **P-network** $\mathrm{P_\theta(s, k)}$: generates continuous parameters $\mathrm{x_k}$ conditioned on the state and a candidate discrete action *k*.

- **Q-network** $\mathrm{Q}_{\omega}(\mathrm{s},\mathrm{x})$: evaluates action values and returns a vector of Q-values across discrete actions, given the state and the continuous parameter vector.

In the implemented design, the P-network input concatenates the 209-dimensional state with a one-hot encoding of the discrete action (21), yielding 230 inputs and producing 10 normalized parameters. The Q-network input concatenates the state (209) with the continuous parameter vector (10), yielding 219 inputs and outputting 21 Q-values. This structure follows the P-DQN framework for discrete–continuous hybrid control [34].

4.3.2 Q-Network Loss Function

The Q-network is trained using a temporal-difference (TD) target consistent with Deep Q-learning:

$$\mathrm{y_t} = \mathrm{r_t} + \gamma \max_{\mathrm{k'}\in\mathcal{K}} \mathrm{Q_{\omega^-}}\left(\mathrm{s_{t+1}}, \mathrm{x'_{k'}}\right), \quad \mathrm{x'_{k'}} = \mathrm{P_{\theta^-}}(\mathrm{s_{t+1}}, \mathrm{k'}), \tag{37}$$

where (ω^{-}, θ^{-}) denote target-network parameters. The Q-loss is:

$$\mathcal{L}Q(\omega) = \mathbb{E}\left[\ell\left(y_t - Q\omega(\mathrm{s_t}, \mathrm{x_t})\right)\right], \tag{38}$$

with $\ell(\cdot)$ typically chosen as Huber loss for robustness. The use of TD learning and target networks is grounded in the original DQN formulation [6].

4.3.3 P-Network Loss Function

The P-network is trained to output parameters that maximize the action value produced by the Q-network. In PDQN, this corresponds to maximizing:

$$\mathrm{J}(\theta) = \mathbb{E}\left[\mathrm{Q_\omega}\left(\mathrm{s}, \mathrm{P_\theta}(\mathrm{s},\mathrm{k})\right)\right], \tag{39}$$

which yields the deterministic policy gradient-style update:

$$\nabla_\theta J(\theta) = E\left[\nabla_x Q_\omega(s,x)\big|_{\,x=P_\theta(s,k)}\, \nabla_\theta P_\theta(s,k)\right]. \tag{40}$$

This gradient form is consistent with deterministic policy gradient theory and its deep instantiation in DDPG, and it is the conceptual basis used when PDQN couples a parameter generator with a critic [43] [34]. In the implemented training description, the same objective is expressed equivalently as minimizing the negative Q-value:

$$\mathcal{L}_{P(\theta)} = -Q_{\omega}\big(\mathrm{s},,\mathrm{k},\mathrm{P}_{\theta}(\mathrm{s},\mathrm{k})\big). \tag{41}$$

4.4 Proposed PDQN-CLR Architecture

This section specifies the PDQN-CLR learning stack used for post-disaster DER dispatch: (i) a dual-network PDQN for hybrid action selection, (ii) a constraint-handling module that introduces adaptive penalties through Lagrange multipliers, and (iii) a safe execution layer that enforces feasibility before OpenDSS execution. Figure 4.1 illustrates the resulting closed-loop system architecture, highlighting the interaction between the PDQN-CLR agent, the safe execution layer, and the OpenDSS-based environment.

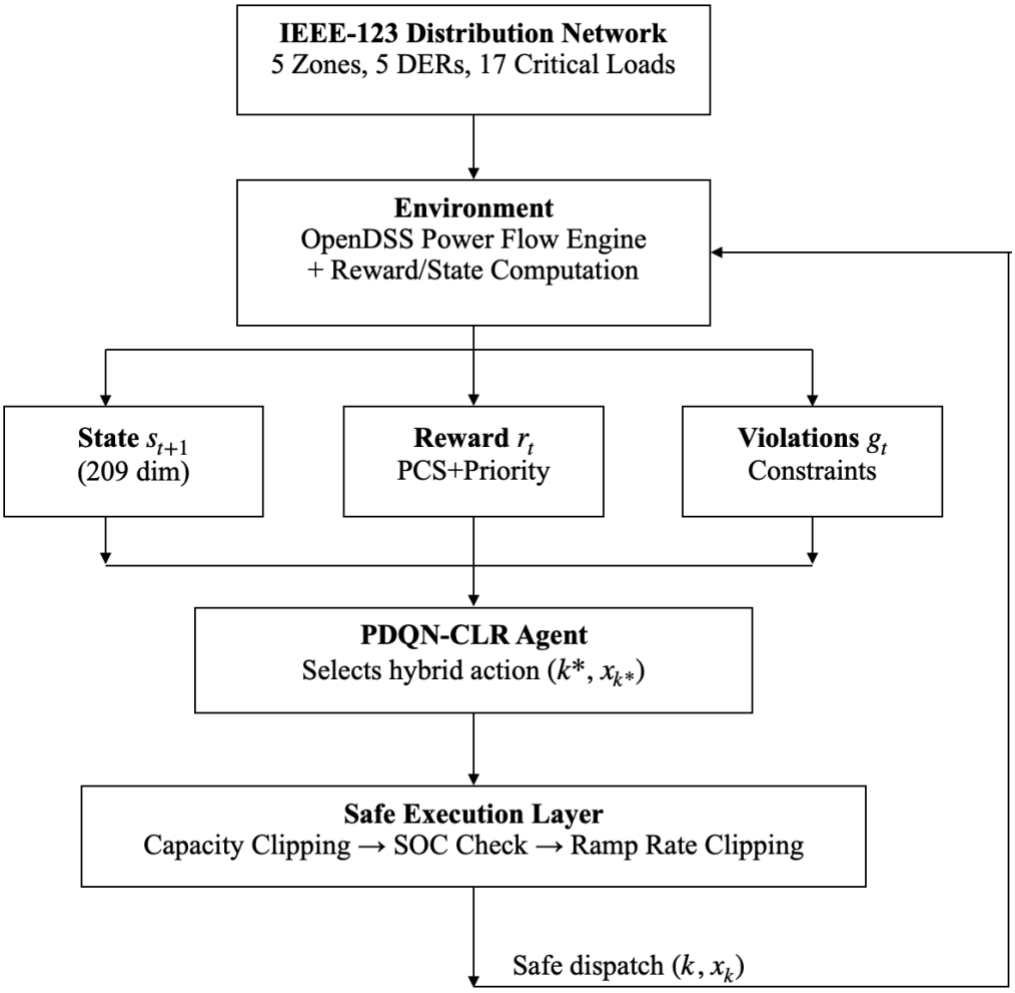

Figure 4.1: PDQN-CLR System Architecture

4.4.1 Q-Network and P-Network Design

PDQN-CLR follows the parameterized action setting, where the executed action at time t is a hybrid pair:

$$\mathrm{a_t} = \left(\mathrm{k_t}, \mathrm{x}_{k_t}\right), \qquad k_t \in \{1, \dots, K\}, \qquad x_{\mathrm{k}} \in \mathbb{R}^{10}, \tag{42}$$

with K=21 discrete action types and $\mathrm{x_k}$ providing continuous DER setpoints (active/reactive for five DERs). The implementation evaluates all discrete actions by

generating x_k per action and scoring each pair (k, x_k), then selecting $k^* = \arg\max_k Q(s, k, x_k)$.

Network roles.

- The parameter network $P(\theta)$ produces action-conditional parameters $x_k = P(s, k; \theta)$ and is called once per discrete action (21 forward passes per decision step).
- The Q-network $Q(\omega)$ scores each candidate hybrid action and outputs 21 Q-values (one per discrete action). To enable gradients from *Q* back to *P*, the critic input concatenates the state with the continuous parameters.

Table 4.4: Network architecture used in PDQN-CLR

Network	Input	Hidden layers	Output	Output bounds
Q-network $Q(s,\cdot)$	209 + 10 = 219	$256 \rightarrow 256$ (ReLU, Dropout 0.1)	21	unbounded Q-values
P-network $P(s,k)$	209 + 21 = 230	$256 \rightarrow 128$ (ReLU, Dropout 0.1)	10	Sigmoid [0,1]

Action-conditioned parameterization.

The one-hot encoding of *k* in the P-network input is the mechanism that makes x_k depend on which discrete action is being considered (e.g., dispatch DER3 vs dispatch DER1 should not share a single continuous optimum).

Normalized-to-physical dispatch mapping.

The P-network outputs normalized parameters in [0,1]. These are scaled to physical DER setpoints by an action manager prior to OpenDSS execution, using (i) a scenario-dependent capacity factor α, and (ii) floor/ceiling limits that restrict the effective dispatch range.

For DER *j*, the active-power scaling is:

$$\boldsymbol{P}_{\mathrm{j}}^{\mathrm{cmd}} = \left(p_{\min} + p_j{}^{\mathrm{norm}}\,(p_{max} - p_{\min})\right)\alpha P_j{}^{\max}, \tag{43}$$

where $p_{\min} = 0.10,\ p_{max} = 0.80$, $p_j^{\mathrm{norm}} \in [0,1]$ and $\alpha \in [0.11,0.55]$ depending on the resource scenario.

Reactive power is scaled bidirectionally as:

$$\mathbf{Q}_{\mathrm{j}}^{\mathrm{cmd}}\left(2\mathrm{q}_{\mathrm{j}}^{\mathrm{norm}} - 1\right)\mathrm{Q}_{\mathrm{j}}^{\max}, \quad \mathrm{q}_{\mathrm{j}}^{\mathrm{norm}} \in [0,1], \tag{44}$$

mapping [0,1] to [-1,1] before applying $\mathrm{Q}_{\mathrm{j}}^{\max}$.

4.4.2 Lagrangian-Based Constraint Handling in The CLR Module

Post-disaster critical load restoration is naturally constrained by operational limits. In constrained RL, these limits are commonly expressed as expected-cost constraints in a constrained Markov decision process (CMDP). The PDQN-CLR design adopts an adaptive Lagrangian penalty approach, conceptually aligned with multiplier-based constraint methods and multi-timescale updates used in modern constrained RL [47].

4.4.2.1 Constraint Formulation

PDQN-CLR tracks three constraint families during training—voltage quality, DER capacity, and dispatch ramping—and evaluates violations from OpenDSS power-flow outcomes at each step.

Let $\mathrm{V_i}$ denote bus voltage magnitudes in per-unit and $\mathrm{P_{j,t}}$ denote DER real power at step *t*. The constraint set is:

- Voltage limits:

$$0.85 \le \mathrm{V_i} \le 1.15, \quad \forall \mathrm{i}. \tag{45}$$

- Capacity limits:

$$0 \le \mathrm{P_{j,t}} \le \alpha \mathrm{P_j^{max}}, \quad \forall \mathrm{j}. \tag{46}$$

- Ramp limits:

$$\left|\mathrm{P_{j,t}} - \mathrm{P_{j,t-1}}\right| \le \rho \mathrm{P_j^{max}}, \quad \forall \mathrm{j}, \tag{47}$$

with $\rho = 0.05$ per step in the reported configuration.

A compact violation construction used by the implementation is:

$$V_t^{\mathrm{viol}} = \sum_i \left([\, V_i - 1.15 \,]_+ + [\, 0.85 - V_i \,]_+ \right) \tag{48}$$

$$C_t^{\mathrm{viol}} = \sum_j \left[P_{j,t} - \alpha P_j^{\max} \right]_+, \tag{49}$$

$$R_t^{\mathrm{viol}} = \sum_j \left[\left| P_{j,t} - P_{j,t-1} \right| - \rho P_j^{\max} \right]_+, \tag{50}$$

where $[\mathrm{z}]_+ = \max(\mathrm{z}, 0)$. The design uses a squared penalty for capacity exceedance to make large over-commands disproportionately expensive.

Constraint Violation Score (CVS).

To summarize operational feasibility over an episode, the Constraint Violation Score (CVS) quantifies the average magnitude of constraint violations per decision step. Let (T) denote the episode horizon here (T=72). At each step (t), the environment computes a nonnegative violation measure $g(t) \ge 0$ by summing squared violation magnitudes from the solved OpenDSS operating point and any residual DER-side exceedances (if present). The episode CVS is then defined as:

$$\mathrm{CVS} = \frac{1}{\mathrm{T}} \sum_{\mathrm{t}=1}^{\mathrm{T}} \mathrm{g(t)}\,. \tag{51}$$

A standard decomposition of g(t) is $\mathrm{g(t)} = \mathrm{V^{viol}(t)} + \mathrm{C^{viol}(t)} + \mathrm{D^{viol}(t)}$,

where $V^{\text{viol}}(t)$ aggregates squared voltage-bound deviations across buses, $C^{\text{viol}}(t)$ captures squared normalized exceedance of any aggregate capacity constraint, and $D^{\text{viol}}(t)$ sums squared normalized exceedances across individual DER limits (when applicable). By construction, $\text{CVS} \in [0, \infty)$, with CVS = 0 indicating fully feasible operation throughout the episode and larger values indicating more frequent and/or more severe violations.

4.4.2.2 Lagrangian Multiplier Integration

Rather than assigning fixed penalty weights, PDQN-CLR augments the step reward using adaptive multipliers $\lambda_v, \lambda_c, \lambda_r$, producing a Lagrangian-shaped objective:

$$\mathbf{r_t^{final}} = r_t^{\text{base}} - \lambda_v, V_t^{\text{viol}} - \lambda_c, \left(C_t^{\text{viol}}\right)^2 - \lambda_r R_t^{\text{viol}}, \tag{52}$$

with λ values bounded to [0,10] in the implementation. This structure parallels the standard CMDP-to-Lagrangian reduction that motivates practical constrained RL methods, including CPO as a reference point for constrained optimization baselines, and RCPO-style multiplier updates as a widely used Lagrangian training pattern [47].

4.4.2.3 Adaptive Multiplier Update Mechanism

PDQN-CLR updates each multiplier based on recent violation history, using an exponential moving average and a tolerance target:

$$\lambda \leftarrow \Pi_{[0,10]}\left(\lambda + \eta_\lambda\left(\hat{\mathrm{g}} - \mathrm{g}^{\text{tol}}\right)\right), \tag{53}$$

where $\hat{\mathrm{g}}$ is an Exponential Moving Average (EMA) of the corresponding violation signal and $\Pi_{[0,10]}$ denotes projection to the configured bounds. The multiplier initialization is selected to reflect the relative operational risk of each constraint, with a stronger initial

emphasis on capacity violations than on ramp-rate violations. The multipliers are initialized as $\lambda_v(0) = 0.5, \quad \lambda_c(0) = 1.0, \quad \lambda_r(0) = 0.01.$

Relation to safe RL alternatives.

Lagrangian penalties are one practical family among several. For context, policy-optimization methods, such as CPO target constraint satisfaction during policy updates, while Lyapunov-based formulations impose safety through Lyapunov conditions; both provide useful contrasts when positioning PDQN-CLR's simpler multiplier-based approach [47]. A broader view of safe RL approaches is summarized in survey literature [48].

4.4.3 Safe Execution Layer

Even with adaptive penalties, exploratory actions may produce infeasible DER setpoints. PDQN-CLR therefore applies a pre-execution feasibility filter that projects the commanded action into admissible operating ranges before solving the power flow. This design is consistent with the safe-RL safety layer concept, where actions are corrected to satisfy constraints prior to environment execution.

Let $\text{clip}(x, \ell, u) = \min\{\max\{x, \ell\}, u\}$. Given commanded active and reactive setpoints $\{P_{j,t}^{cmd}, Q_{j,t}^{cmd}\}_{j=1}^{5}$, the safe execution layer enforces feasibility through the following sequence.

Step 1: Capacity and reactive-power limits.

$$P_{j,t}^{(1)} = \text{clip}\left(P_{j,t}^{cmd}, 0, \alpha P_j^{max}\right), \qquad Q_{j,t}^{(1)} = \text{clip}\left(Q_{j,t}^{cmd}, -Q_j^{max}, Q_j^{max}\right). \tag{54}$$

Step 2: State-of-charge reserve enforcement.

A minimum reserve $SOC_{\min} = 0.10$ is enforced. If $SOC_{j,t} < SOC_{\min}$, dispatch from DER *j* is blocked (or reduced) to preserve the reserve margin:

$$P_{j,t}^{(2)} = \begin{cases} 0, & SOC_{j,t} < SOC_{min}, \\ P_{j,t}^{(1)}, & SOC_{j,t} \geq SOC_{min}. \end{cases} \quad (55)$$

Step 3: Ramp-rate limiting:

$$P_{j,t}^{(3)} = clip\left(P_{j,t}^{(2)},\ P_{j,t-1} - \rho\alpha P_j^{max},\ P_{j,t-1} + \rho\alpha P_j^{max}\right), \rho = 0.05. \quad (56)$$

The filtered setpoints $\{P_{j,t}^{(3)}, Q_{j,t}^{(3)}\}$ are then passed to OpenDSS for evaluation. In practice, as training progresses and the policy becomes more constraint-aware, the frequency and magnitude of these corrective projections typically decrease.

4.5 Training Methodology

PDQN-CLR is trained off-policy using experience replay, target networks, and an exploration schedule consistent with value-based deep RL practice [49]. To improve clarity and reproducibility, Algorithm 4.1 summarizes the core PDQN-CLR training loop. At each time step, the agent evaluates the discrete action candidates using action-conditioned parameter generation, applies ε-greedy exploration, and projects the selected continuous setpoints through the safe execution layer before running the OpenDSS power-flow step. The resulting restoration metrics are converted into a shaped reward with explicit constraint-violation signals, after which the Q- and P-networks are updated off-policy using prioritized replay, target networks (soft updates), and adaptive multiplier updates for constraint enforcement. This procedure follows the standard P-DQN training template and incorporates the CLR and safety mechanisms introduced in Section 4.4.2–

4.4.3; the remainder of Section 4.5 specifies the replay, target-tracking, and exploration settings used in the experiments.

Algorithm 1: PDQN-CLR (Core Training Procedure)

Result: Trained PDQN-CLR agent with safety constraints

Initialize online networks (Q_ω, P_θ), target networks (Q_{ω^-}, P_{θ^-}), replay buffer $\mathcal{D}$ (PER), and CLR multipliers λ (bounded);
for *episode* = 1 **to** E **do**
  Reset OpenDSS and observe s_0;
  for $t = 0$ **to** $T - 1$ **do**
    $\epsilon \leftarrow$ DecayEpsilon();
    $(k, x) \leftarrow$ SelectPDQNAction$(s, \epsilon; Q_\omega, P_\theta)$;
    $\tilde{x} \leftarrow$ SafetyFilter(x);
    $(s', r_{\text{base}}, g, \text{done}) \leftarrow$ OpenDSS_Step$(k, \tilde{x})$;
    $r \leftarrow$ ShapeReward$(r_{\text{base}}, g, \lambda)$;
    StorePER$(\mathcal{D}, (s, k, \tilde{x}, r, g, s', \text{done}))$;
    $\lambda \leftarrow$ UpdateMultipliers(λ, g);
    if *ReadyToLearn*$(\mathcal{D})$ **then**
      PER_Update$(Q_\omega, P_\theta, Q_{\omega^-}, P_{\theta^-}, \mathcal{D})$;
    end
    SoftUpdateTargets$(Q_{\omega^-}, P_{\theta^-}; Q_\omega, P_\theta)$;
    $s \leftarrow s'$;
    if *done* **then**
      break;
    end
  end
end

4.5.1 Prioritized Experience Replay

Instead of uniform replay sampling, PDQN-CLR uses prioritized experience replay (PER), where transitions with larger TD error are sampled more often [50]. The configured priority and correction terms follow:

$$\mathrm{p_i} = (|\delta_\mathrm{i}| + \varepsilon)^{\alpha_{\mathrm{PER}}}, \quad \mathrm{P(i)} = \frac{\mathrm{p_i}}{\sum_\mathrm{j} \mathrm{p_j}}, \quad \mathrm{w_i} = \frac{\big(\mathrm{N}, \mathrm{P(i)}\big)^{-\beta}}{\max_\mathrm{j}\big(\mathrm{N}\ \mathrm{P(j)}\big)^{-\beta}}, \tag{57}$$

With $\alpha_{PER} = 0.6$, β: 0.4 → 1.0 annealed over 100,000 steps, and $\varepsilon = 10^{-6}$.

4.5.2 Target Network Updates

To stabilize TD targets, PDQN-CLR maintains target networks (θ^-, ω^-) and performs a soft update:

$$\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-, \qquad \omega^- \leftarrow \tau\omega + (1-\tau)\omega^-,$$

With $\tau = 0.005$. This update mirrors the target network idea that underpins stable Deep Q-learning and is widely used in actor–critic practice for smooth tracking [49].

4.5.3 Exploration strategy with epsilon decay

PDQN-CLR applies an ε-greedy schedule over discrete actions: with probabilityε, a random discrete action *k* is selected; otherwise *k* is chosen greedily by the critic. This is the standard exploration mechanism described in reinforcement learning texts and is commonly used with DQN-style methods [49].

The exploration schedule is set to:

$$\varepsilon_{\text{start}} = 1.0, \quad \varepsilon_{\text{end}} = 0.01,$$

with linear decay over approximately 2,800 episodes (~201,600 steps).

4.5.4 Hyperparameter selection

Table 4.5 summarizes the key training hyperparameters used for PDQN-CLR, drawn from the project configuration.

Table 4.5: PDQN-CLR training hyperparameters

Item	Value	Item	Value
Learning rate *Q*	10^{-4}	Soft target update τ	0.005
Learning rate *P*	10^{-4}	ε-greedy	1.0 → 0.01 over ~2,800 episodes
Discount factor γ	0.99	α_{PER}	0.6

Item	Value	Item	Value
Batch size	64	PER β	0.4 → 1.0 (100,000 steps)
Replay buffer size	100,000	PER ε	10^{-6}

Because deep RL outcomes can vary substantially with implementation details and hyperparameter choices, reporting these settings explicitly follows widely cited guidance on evaluation discipline and reproducibility in deep RL studies [51].

4.5.5 Computational Requirements

Training was performed on Oakland University's high-performance computing (HPC) cluster using an NVIDIA Tesla V100 PCIe (16 GB). A complete training run of 3,000 episodes (216,000 environment steps) finished in approximately 0.89 h, with peak GPU memory usage of approximately 408 MiB. Table 4.6 reports the training hardware and runtime statistics. The observed GPU utilization was modest (≈3–20%), which is attributed to the per-step OpenDSS power-flow evaluation dominating wall-clock time relative to the neural-network forward/backward passes. OpenDSS's standard engine is implemented as a CPU-based simulator (with sparse linear solves via KLU and CPU-oriented parallel features, such as Actors), so the power-flow call remains the principal runtime driver in the closed-loop training pipeline.

Table 4.6: Training hardware specifications

Component	Specification
Platform	Oakland University HPC
GPU	NVIDIA Tesla V100-PCIE-16GB
GPU Memory Used	~408 MiB

Component	Specification
Training Time	0.89 hours (3,000 episodes)
Total Training Steps	216,000
Inference Time	<1 ms per action

Per-step timing profile and dominant cost. In addition to the aggregate runtime statistics in Table 4.6, step-level profiling was performed to identify the dominant contributors to wall-clock time during closed-loop training. Each environment step performs one OpenDSS solve (Snapshot; see Section 3.1.3). Profiling in the reported configuration indicates an average OpenDSS power-flow solve time of approximately 5 ms per decision step, while policy evaluation (neural-network forward pass for action selection) remains sub-millisecond (<1 ms per action). Over the fixed 72-step horizon, this yields an episode-level simulation time of ~0.36 s (≈72 × 5 ms) attributable to the power-flow solves alone, indicating that simulator-in-the-loop evaluation, rather than neural computation, is the principal throughput limiter.

Table 4.7: Runtime breakdown per decision step

Component (per decision step)	Typical time	Notes
OpenDSS Snapshot power-flow solve	~5 ms	One solve per step in Snapshot mode
Policy forward pass (inference)	<1 ms	Table 4.6 reports sub-ms inference
Episode length	72 steps	Fixed 36-hour horizon at 30-min steps (Chapter Three)

Scalability considerations. The reported timings reflect the adopted IEEE 123-bus configuration and the fixed 72-step horizon. As system size grows, the dominant contributor in simulator-driven training remains the per-step power-flow solve and

associated data movement, consistent with the throughput limitation noted in the dissertation's limitations discussion. Accordingly, scaling to larger feeders and/or longer horizons primarily increases the offline training cost through more expensive environment steps, whereas the learned policy's online inference remains lightweight relative to power-flow evaluation and is well-suited to time-critical restoration decision loops emphasized in resilience-oriented control studies.

4.6 Chapter Summary

This chapter section detailed the PDQN-CLR methodology beyond the base PDQN: the dual-network architecture and parameter scaling used to map normalized outputs to DER setpoints, the CLR module that enforces voltage, capacity, and ramp constraints through adaptive Lagrangian multipliers, and the safe execution layer that clips actions before OpenDSS simulation. The training pipeline combines PER, target networks with soft updates, and a decaying ε-greedy exploration schedule, yielding a practical constrained hybrid-control framework aligned with established safe RL concepts while remaining computationally feasible for repeated OpenDSS power-flow evaluation.

CHAPTER FIVE

RESULTS

Chapter five presents the empirical evaluation of PDQN-CLR on the post-disaster islanded restoration problem defined earlier in the dissertation, with emphasis on both restoration quality (priority-weighted service) and operational feasibility (constraint satisfaction). Two resource regimes are considered—Sufficient (CF≈50%) and Scarce (CF≈12%)—so that the reported trends reflect both capacity-abundant and capacity-limited conditions. The chapter is organized to move from (i) reporting conventions and metric definitions (Section 5.1), to (ii) evidence of stable learning dynamics across 3,000 training episodes (Section 5.2), and then to (iii) the primary performance comparison against a Greedy baseline (Section 5.3). Subsequent sections explain why the headline gains occur by decomposing outcomes into tier-level service allocation (Section 5.4), and how the policy achieves them through step-wise dispatch behavior over the 72-step horizon (Section 5.5), before quantifying how both PCS and CVS scale with available resources (Section 5.6).

5.1 Evaluation Protocol and Reporting Conventions

This section specifies the evaluation protocol and reporting conventions used throughout this chapter. It summarizes the elements required to interpret the results—namely the resource regimes considered, the compared baselines, and the definitions of the reported metrics and statistical summaries—so that the training, comparison, and ablation figures in Sections 5.2–5.6 can be read on a common basis.

5.1.1 Evaluation Scope and Resource-Availability Regimes

All evaluations are conducted in the same closed-loop simulation environment, where system transitions are produced by OpenDSS power-flow execution under islanded operation.

Two resource regimes are considered:

- **Sufficient resources**: capacity factor centered at CF ≈ 50% (implemented as a small range around the nominal value).
- **Scarce resources**: capacity factor centered at CF ≈ 12% (implemented as a small range around the nominal value).

The capacity factor models post-disaster derating of DER real-power capability:

$$P_{available} = \alpha P_{rated}, \tag{58}$$

where $\alpha \in [0,1]$ is the capacity factor.

Each episode spans a fixed restoration horizon of 72 steps at 30 minutes per step (36 hours total).

To prevent overfitting to a single disturbance configuration, evaluation episodes follow the same stochastic sampling conventions used in the environment definition (fault scenario, capacity factor, and initial SOC sampling).

Load service credit is computed at every step using the hybrid service-quality test described in Chapter 4 (voltage admissibility plus DER-support attribution for electrically distant buses), rather than a voltage-only rule that can be biased by slack-source masking.

5.1.2 Baselines Compared

Two methods are compared:

1. PDQN-CLR (proposed): the trained hybrid-action agent described in Chapter 4, combining (i) discrete–continuous action selection in the P-DQN family, (ii) adaptive constraint penalties, and (iii) pre-execution feasibility enforcement through the safe execution layer.
2. Greedy (baseline): a deterministic, non-learning, step-by-step heuristic executed in the same OpenDSS environment and evaluated under the same episode sampling and service quality tests as PDQN-CLR. At each step, it restores unserved loads using a fixed priority order (P1 → P2 → P3) and sets the dispatch parameters using a simple rule-based allocation (e.g., nearest-DER dispatch). It performs no value-function learning, no replay-based learning, no exploration schedule, no adaptive constraint-penalty updates, and no multi-step credit assignment; therefore, decisions are myopic and do not improve with additional episodes.

Table 5.1: PDQN-CLR vs. Greedy baseline

Feature	PDQN-CLR	Greedy (baseline)
Policy representation	Neural function approximators (Q-network + parameter network for hybrid actions)	None (rule-based heuristic)
Value-function learning	Temporal-difference learning with target-network stabilization	None
Experience replay	Prioritized experience replay (buffered transitions; prioritized sampling)	None
Exploration	ε-greedy exploration (and parameter perturbation during exploration)	Deterministic only (no exploration)

Feature	PDQN-CLR	Greedy (baseline)
Constraint handling during learning	Adaptive constraint-penalty mechanism (Lagrangian-style multipliers)	None (no adaptive penalty mechanism)
Action selection principle	Learned scoring of discrete templates + learned continuous setpoints	Fixed priority heuristic for restoration choices
Multi-step planning	Discounted return with multi-step credit assignment (Bellman backups)	Single-step, myopic decisions
Optimization	Gradient-based training (e.g., Adam)	None
State processing	High-dimensional state feature encoding used by networks	Minimal state extraction for heuristic rules
Temporal credit assignment	Yes (future effects influence current updates through return estimation)	None

Unless otherwise stated, reported results are computed over 150 evaluation episodes per regime.

5.1.2.1 Baseline Selection Rationale

The Greedy heuristic is used as the primary baseline because it provides a transparent, deterministic point of comparison under the same simulator-in-the-loop evaluation protocol, while avoiding the additional modeling assumptions required by optimization-based benchmarks. In the implemented setting, the controller operates over a hybrid action space with 21 discrete templates and 10 continuous DER setpoints per step, evaluated over a fixed 72-step horizon.

A direct comparison against MPC/MILP-style controllers is not included because it would require formulating and repeatedly solving a mixed-integer optimization problem coupled to a three-phase distribution power-flow model. In practice, many

restoration-oriented optimization studies rely on linearized three-phase power-flow approximations and then validate feasibility in OpenDSS, highlighting that such baselines are not strictly model-matched to the nonlinear simulator used here. In addition, reported computation times for MILP-based critical-load restoration are commonly on the order of seconds to tens of seconds per instance for benchmark feeders, even before accounting for repeated solves across scenario sampling and sensitivity sweeps. By contrast, the proposed evaluation pipeline is designed around high-throughput sampling, where each environment step executes a single OpenDSS snapshot solve (~5 ms) and policy inference remains sub-millisecond, enabling large-scale rollouts (e.g., 3,000 episodes) within practical runtimes.

Finally, the Greedy baseline establishes a clear minimum-viable performance reference that reflects common engineering intuition—priority-ordered restoration and simple rule-based dispatch—without learning, adaptive constraint penalization, or multi-step credit assignment. This makes improvements attributable to PDQN-CLR's learned policy and constraint-handling mechanisms easier to interpret than comparisons against heavily tuned optimization objectives.

5.1.3 Metrics and Statistical Reporting

Three primary outcome measures are reported: Priority-Weighted Critical Score (PCS), Constraint Violation Score (CVS), and tier service rates (i.e., per-tier service consistency under the three-tier critical-load structure P1/P2/P3).

(a) Priority-Weighted Critical Score (PCS).

PCS is computed at each decision step using the definition formalized in chapter four. In brief, each critical load is assigned a binary service indicator under the hybrid

service evaluation (voltage admissibility plus DER-support attribution for electrically distant buses), and these indicators are aggregated using the priority-tier weights. The tier definitions and associated weights follow Table 3.3.

(b) Constraint Violation Score (CVS).

The feasibility is summarized using the Constraint Violation Score (CVS) defined in Section 4.4.2.1. In brief, the CVS captures the episode-average magnitude of post-execution constraint violations across the restoration horizon based on the monitored operating-point limits (e.g., voltage-limit violations observed in the OpenDSS solution, and any residual infeasibility not eliminated by the pre-execution filtering stage). PCS (and restored-load counts) is computed from service indicators defined only on the 17 critical-load buses under the hybrid service evaluation described in this section, whereas CVS aggregates operating-point constraint deviations over the full feeder at each step and then averages them over the T-step episode. Since the voltage-violation term sums deviations across buses ($\forall i$) in the solved OpenDSS state, CVS can increase even when the restored critical-load count is unchanged: the Greedy baseline may keep the critical-load buses within the service test while still producing off-limit voltages elsewhere on the feeder and/or short-lived excursions during the horizon, which do not necessarily change the binary critical-load service indicators used in PCS. A CVS value of 0 indicates fully feasible operation throughout the episode, whereas larger values indicate more severe and/or more persistent violations.

(c) Tier service rates (SR).

Tier service rates quantify the temporal consistency of restoration within each priority tier. For each critical load $i \in C$, a binary service indicator $s_i(t) \in \{0,1\}$ is

recorded at each decision step t using the same hybrid service evaluation applied for PCS. The per-load service rate over an episode of length T is defined as:

$$\text{ServiceRate}_{\text{i}} = \frac{1}{T}\sum_{t=1}^{T} s_i(t)\,. \tag{59}$$

The tier service rate is then computed as the arithmetic mean of per-load service rates within tier k:

$$\text{TierServiceRate}_{\text{k}} = \frac{1}{|L_k|}\sum_{\text{i}\in\text{L}_{\text{k}}} \text{ServiceRate}_{\text{i}}\,, \tag{60}$$

where $L_k \subseteq C$ denotes the set of critical loads assigned to priority tier k. For reporting and visualization, $\text{TierServiceRate}_{\text{k}}$ is expressed as a percentage, i.e., $100\times\text{TierServiceRate}_{\text{k}}$. Unlike PCS, which aggregates service using tier weights, tier service rates provide an unweighted, disaggregated view of service consistency and therefore make scarcity-driven trade-offs across tiers explicit.

(d) Statistical Summaries and Plot Conventions

For any episode-level metric m evaluated over n episodes, let $\text{m}^{(\text{e})}$ denote the value from episode $\text{e} \in \{1, \ldots, \text{n}\}$. The reported mean is:

$$\bar{\text{m}} = \frac{1}{\text{n}}\sum_{\text{e}=1}^{\text{n}} \text{m}^{(\text{e})}\,. \tag{61}$$

Dispersion across episodes is summarized using the standard deviation computed with divisor (n) (consistent with the default behavior of $np.std(\cdot)$ with $ddof = 0$:

$$\sigma_{\text{m}} = \sqrt{\frac{1}{\text{n}}\sum_{\text{e}=1}^{\text{n}} (\text{m}^{(\text{e})} - \bar{\text{m}})^2}\,. \tag{62}$$

In the evaluation bar plots, bars show $\bar{m}$ and error bars show $\pm\sigma_m$ across evaluation episodes. For training curves, the solid line shows the reported average trend, and shaded bands indicate variability as specified in the figure caption (e.g., across episodes/runs or within a smoothing window).

5.2 Training Convergence and Stability

This section evaluates whether PDQN-CLR training exhibits stable convergence under both resource regimes over 3,000 episodes.

5.2.1 Episode Reward Convergence

Figure 5.1(a) shows that episode return increases rapidly during early training and stabilizes well before the end of training in both regimes. Under sufficient resources, the converged return is approximately 4.4×10^3, whereas under scarcity it stabilizes near 3.9×10^3. This separation is expected because the scarce regime imposes a physical limit on achievable restoration, reducing the maximum attainable PCS and, consequently, cumulative reward.

5.2.2 PCS Convergence

Figure 5.1(b) reports convergence of PCS, the primary restoration objective. In the sufficient regime, PCS stabilizes in the range 0.92–0.94, approaching full restoration. In the scarce regime, PCS stabilizes around 0.78–0.80, consistent with the theoretical upper bound implied by the weighted restoration ceiling under severe derating.

5.2.3 Constraint Behavior During Training and Exploration Decay

Figure 5.1(c) focuses on early-phase constraint satisfaction. CVS drops to near zero within the first few dozen episodes in both regimes, which is consistent with the

intended interaction between proactive feasibility enforcement (safe execution) and post-execution penalization (adaptive constraint penalties).

Figure 5.1(d) documents the ε-greedy exploration schedule decaying from 1.0 to 0.01 across training, which ensures early coverage of the action space while transitioning toward exploitation as the value function stabilizes.

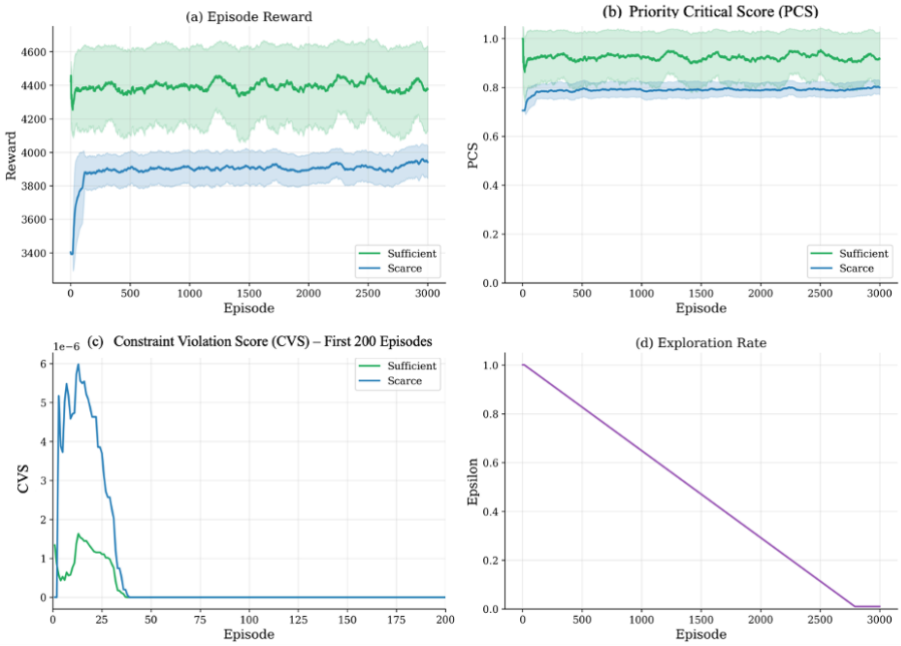

Figure 5.1: Training Dashboard

5.3 PDQN-CLR Performance Evaluation

Figure 5.2 reports the headline comparison between PDQN-CLR and the Greedy baseline under sufficient and scarce resources, using the evaluation protocol in Section 5.1.

5.3.1 PCS Under Sufficient and Scarce Resource Conditions

Under sufficient resources, PDQN-CLR achieves a mean PCS of 0.94, compared with 0.80 for Greedy, corresponding to a 17.5% increase in priority-weighted restoration and a statistically significant improvement over 150 evaluation episodes ($p < 0.01$). Under scarce resources, both methods converge to a similar PCS (≈ 0.82), reflecting the binding nature of the capacity limit in this regime.

5.3.2 CVS Under Sufficient and Scarce Resource Conditions

The most pronounced separation is in constraint satisfaction. PDQN-CLR maintains CVS near 0.009 in both regimes, whereas Greedy exhibits substantially higher violation scores (≈ 0.388 in the sufficient case and 0.667 in the scarce case), corresponding to approximately 43× and 74× reductions, respectively. This separation is expected because Greedy selects setpoints using a fixed priority heuristic and does not incorporate the adaptive constraint-penalty learning and multi-step credit assignment used by PDQN-CLR, and is therefore more prone to repeatedly proposing operating points that trigger network-limit violations. Notably, this CVS separation can coexist with similar restored-load counts under scarcity because CVS captures feeder-wide, time-averaged operating-point deviations, whereas restored-load counts are based on critical-load service indicators.

5.3.3 Feasibility as A Discriminating Criterion Under Restoration-Count Ties

A key interpretation of Fig. 5.2 is that PCS (and even restored-load counts) can mask operational risk. Under the scarce regime, Greedy can approach the same PCS ceiling as PDQN-CLR because restoration is fundamentally constrained by available DER power; however, Fig. 5.2(b) shows that it does so with a substantially higher CVS.

This distinction matters in the present islanded OpenDSS setting because voltage-only indicators can be inflated by slack-source effects, motivating the hybrid service-credit rule used throughout this dissertation.

Accordingly, the main finding from Figure 5.2 is twofold: (i) when resources permit, PDQN-CLR increases priority-weighted restoration relative to a myopic heuristic; and (ii) across all resource regimes, PDQN-CLR produces dispatch decisions that remain close to feasible operation as measured by CVS.

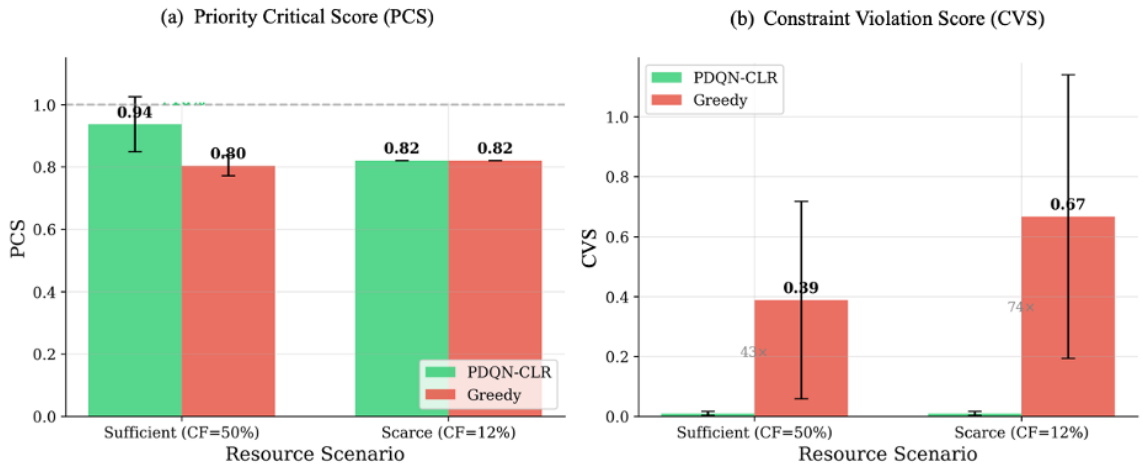

Figure 5.2: PDQN-CLR performance evaluation

5.4 Priority-Aware Restoration Performance

This section decomposes the headline PCS results (Section 5.3) into which tiers are preserved and how restoration is allocated across the three-tier priority structure defined in Chapter 3. The analysis is aligned with the priority-weighted objective used during training (PCS-based shaping) and therefore provides direct evidence that PDQN-CLR's policy internalizes the intended prioritization logic.

5.4.1 Tier Service Rates and The Scarcity Trade-Off

Figure 5.3 shows the tier service rates, defined as the fraction of time steps for which loads in a given tier receive service credit. Under Sufficient resources, PDQN-CLR maintains high service rates across all tiers (P1: 98.4%, P2: 89.1%, P3: 92.5%), indicating that the agent can simultaneously satisfy critical and lower-priority demands when capacity margins are available.

Under Scarce resources, the service profile shifts in a way that matches the intended restoration policy: P1 service increases to 100%, while P2 and P3 decrease to 64.5% and 66.7%, respectively. Quantitatively, the shift corresponds to a prioritization ratio of approximately 16×, computed as the P3 service reduction (92.5%→66.7%, i.e., 25.8 percentage points) divided by the P1 improvement (98.4%→100.0%, i.e., 1.6 percentage points), indicating that P3 service is reduced roughly sixteen times more than P1 as the regime transitions from sufficient to scarce resources. This asymmetric response is consistent with maximizing PCS, because shedding lower-weight loads (P3, ζ=0.4) protects higher-weight loads (P1, ζ=1.0) when the capacity factor forces binding scarcity.

Importantly, this behavior is not imposed by a hand-coded dispatch rule. Instead, it is an outcome of optimizing the reward structure in Chapter 4, where PCS is the primary term and constraint penalties discourage infeasible attempts to serve all tiers simultaneously under scarcity.

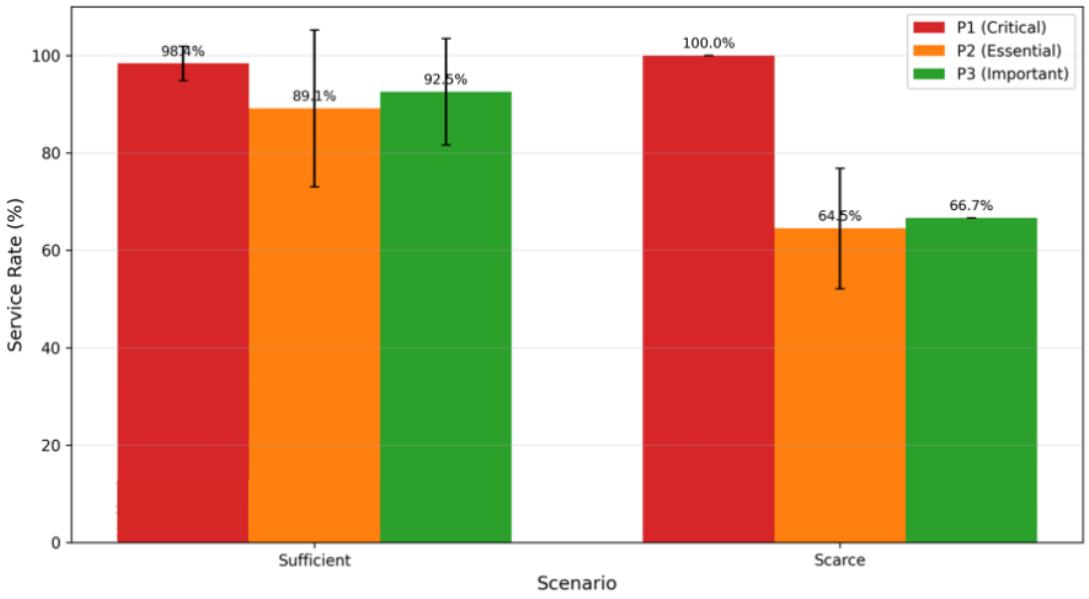

Figure 5.3: PDQN-CLR Priority Service Rates

5.4.2 Tier-Wise Restored Load Counts and Interpretation Under Scarcity

Figure 5.4 complements the service-rate analysis by reporting absolute loads restored by tier, averaged over evaluation episodes (hence fractional values). In the Sufficient regime, PDQN-CLR restores 15.7/17 critical loads on average, compared with 12.8/17 for Greedy. The gain is driven primarily by improved P2 and P3 restoration (PDQN-CLR: P2 ≈ 3.3 vs Greedy ≈ 2.0; P3 ≈ 5.4 vs Greedy ≈ 3.8), while both methods fully serve all seven P1 loads in this regime.

In the Scarce regime, both methods converge to 13.0/17 restored loads. This apparent tie is expected in a capacity-limited setting and should be interpreted as a limitation of quantity-only summaries: when available DER power is far below the level needed for full restoration, multiple strategies can yield similar counts, even if their feasibility differs. For this reason, Fig. 5.4 should be read alongside the feasibility result

in Fig. 5.2: PDQN-CLR achieves similar load counts under scarcity while maintaining low CVS, whereas Greedy's comparable restoration occurs with markedly higher constraint violations.

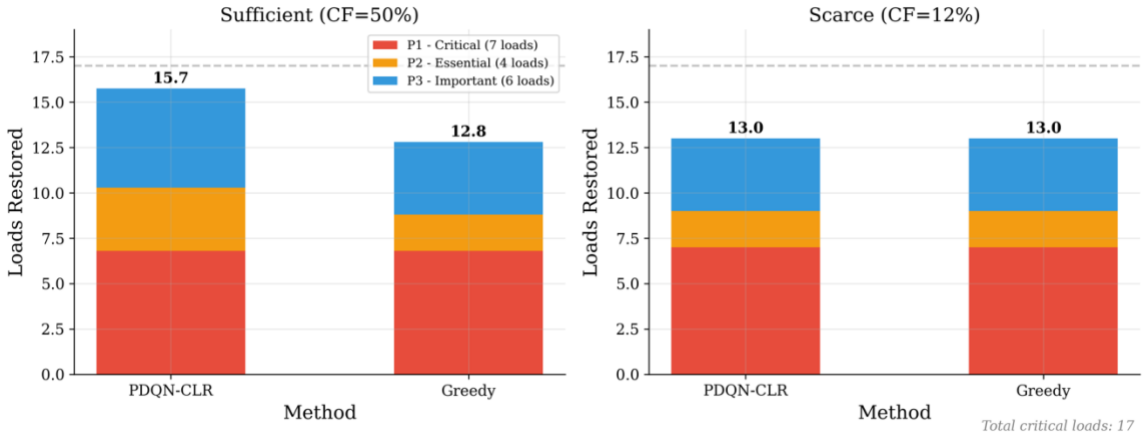

Figure 5.4: Priority-Based Load Restoration

5.5 Behavioral Analysis: Dispatch Trajectories

The previous section established what PDQN-CLR restores in terms of tier allocation. Figure 5.5 explains how these outcomes are realized by reporting the time evolution of DER real-power dispatch (stacked by DER) together with the corresponding PCS trajectory over the 72-step episode horizon, under both resource regimes. The figure is consistent with the closed-loop execution pipeline described in Chapter 4: at each decision step, the controller observes the current state, selects a hybrid action, applies safe-execution filtering, executes an OpenDSS power-flow evaluation, and then updates the service and violation signals used for subsequent decisions.

It is important to note that the common plateau at approximately 76.5% corresponds to the baseline restoration level provided by the islanded source configuration (Vsource), which supplies 13 of the 17 critical loads under the adopted service-credit rule. Both

regimes therefore converge to this baseline later in the horizon, consistent with the effect of intertemporal operating limits (e.g., energy/SOC constraints) that reduce the ability to sustain high-output operation over the full episode.

5.5.1 Sufficient Regime: Dispatch Phases and PCS Transition

In the Sufficient case (CF≈50%), PDQN-CLR sustains a high aggregate dispatch (approximately 250 kW) during the early portion of the episode, with PCS held at 100%, indicating full priority-weighted restoration during this interval. In practical terms, this corresponds to maintaining PCS = 100% for approximately the first 21 steps, i.e., about 10.5 hours of simulated time under the 30-minute step resolution. The dispatch then decreases in a stepwise manner, and PCS exhibits a distinct transition at approximately step ~21, dropping to ≈76.5% and remaining essentially flat thereafter. Over the remainder of the horizon, dispatch stabilizes at a lower level (roughly 120 kW). The stepwise reduction in dispatch is consistent with intertemporal operating limits (e.g., SOC reserve enforcement and energy/ramp constraints) described in Chapter 4; however, because SOC is not shown in Fig. 5.5, this linkage is presented as an interpretation rather than a directly observed mechanism.

5.5.2 Scarce Regime: Lower Dispatch Envelope and Stability

In the Scarce case (CF ≈ 12%), the available capacity is much smaller (≈ 77 kW, dashed line), and PDQN-CLR maintains a comparatively low and stable dispatch envelope (≈60 – 62 kW) over most of the episode. PCS remains approximately constant at ≈76.5% across the horizon, reflecting that the capacity-limited regime compresses the attainable restoration level.

A modern dispatch reduction occurs near step ~43 (to approximately 52 kW), yet PCS remains unchanged, indicating that the reduced dispatch is still sufficient to maintain the achieved service level under the adopted crediting rule.

5.5.3 Interpretation linked to feasibility and prioritization mechanisms

The contrasting trajectories in Fig. 5.5 align with the design elements formalized in Chapter 4:

1. Intertemporal constraint awareness. Because the state representation includes energy-related variables (e.g., SOC) and DER capability bounds, the policy can condition dispatch on remaining energy and operating limits rather than treating dispatch as a static allocation problem.
2. Priority-driven objective shaping. PCS is computed at each step and embedded in the training objective; consequently, under sufficient resources the policy is incentivized to realize high-priority service early (high PCS), even if later reductions become necessary as constraints tighten.
3. Feasibility-oriented execution. The safe execution layer enforces DER-side feasibility prior to OpenDSS evaluation, while network feasibility is assessed after power flow and reflected through the violation signals used during learning and evaluation.

Taken together, these mechanisms explain why PDQN-CLR exhibits resource-adaptive behavior in Fig. 5.5—supporting a finite interval of full restoration when capacity is

sufficient, while maintaining a conservative, stable dispatch profile under scarcity—without requiring explicit, hand-coded mode switching.

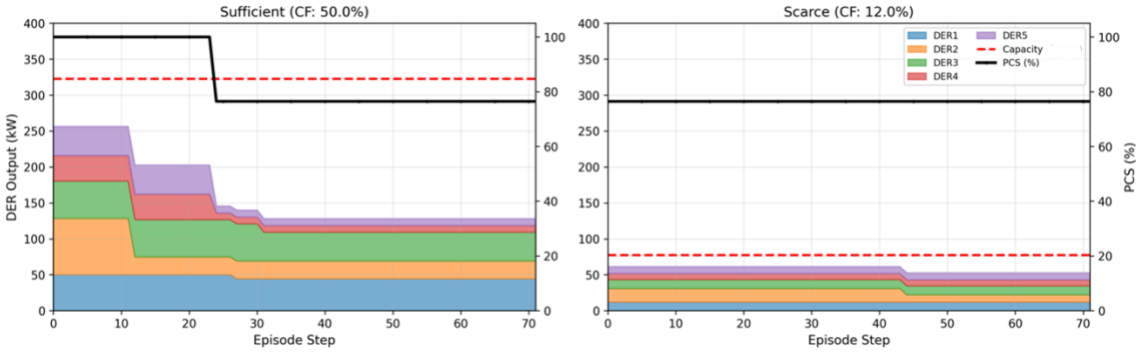

Figure 5.5: PDQN-CLR: DER Dispatch and Load-service Timeline (Sufficient vs. Scarce)

5.6 Sensitivity to Resource Availability

Figure 5.6 evaluates how restoration quality and feasibility change as DER availability increases from CF≈12% to CF≈50%. The goal is to verify that PDQN-CLR (i) exploits additional resources when present and; (ii) maintains low CVS across regimes.

5.6.1 PCS Versus Capacity Factor

Figure 5.6(a) shows that PDQN-CLR exhibits positive scaling with resource availability: PCS increases from approximately 0.82 at CF≈12% to 0.94 at CF≈50%. In contrast, Greedy remains nearly flat (≈0.80–0.82), indicating limited ability to convert added capacity into higher priority-weighted restoration.

This contrast aligns with the behavioral evidence in Fig. 5.5: PDQN-CLR uses higher capacity to temporarily achieve near-complete restoration, while Greedy does not reliably translate increased margins into sustained, priority-consistent service.

5.6.2 CVS Versus Capacity Factor

Figure 5.6(b) indicates that PDQN-CLR maintains consistently low CVS (≈0.01) at both capacity factors, while Greedy's CVS increases sharply under scarcity (≈0.39 at CF≈50% to ≈0.67 at CF≈12%). This pattern supports the main safety claim established in Fig. 5.2: feasibility remains a distinguishing outcome even when restoration scores compress under scarcity.

5.6.3 Resource-Dependent Performance Regimes

The sensitivity analysis highlights a practical threshold effect. When the available DER power is a small fraction of what is needed for full restoration (scarce regime), PCS improvements are bounded and count-based outcomes can tie, but feasibility differences become decisive. As capacity increases toward the sufficient regime, PDQN-CLR converts the added margin into materially higher priority-weighted restoration while maintaining low violations.

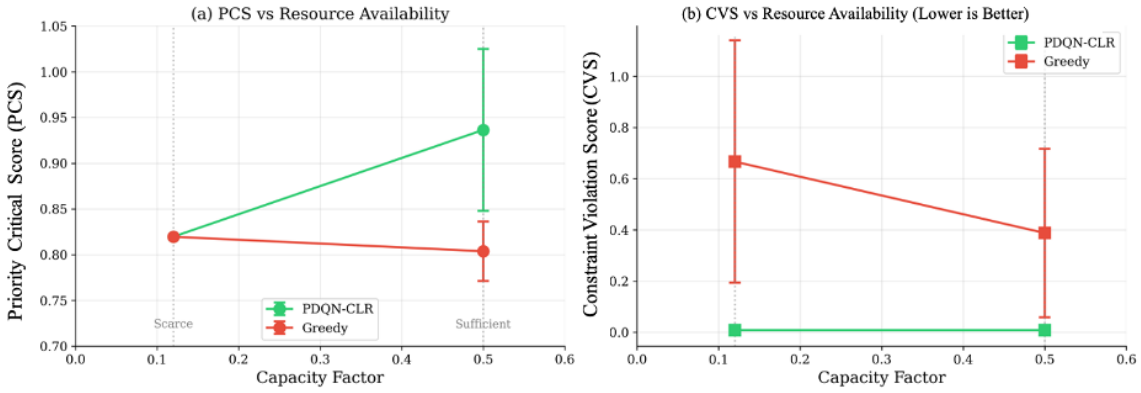

Figure 5.6: Capacity Factor Analysis

5.7 Chapter Summary

This chapter evaluated PDQN-CLR under sufficient and scarce resource regimes. Training results indicate stable convergence in episode return and PCS, with early

reductions in constraint violations (Fig. 5.1). In evaluation, PDQN-CLR improves priority-weighted restoration when resources are sufficient and maintains substantially lower violation scores than the Greedy baseline across regimes (Fig. 5.2). Tier-level analysis shows that, under scarcity, PDQN-CLR preserves P1 service while trading off lower tiers, and highlights that restored-load counts alone can obscure feasibility differences when methods tie on quantity (Figs. 5.3 and 5.4). Dispatch trajectories further demonstrate resource-dependent behavior, with high early dispatch under sufficient resources followed by a stepwise reduction, and a low, stable dispatch profile under scarcity (Fig. 5.5). Finally, sensitivity analysis confirms that PDQN-CLR scales appropriately with increasing capacity, improving PCS while keeping CVS low (Fig. 5.6).

CHAPTER SIX

CONCLUSION AND FUTURE WORK

This chapter concludes the dissertation by consolidating the central outcomes of the proposed PDQN-CLR framework for post-disaster critical load restoration in islanded microgrids. It synthesizes the key empirical findings from the IEEE 123-node OpenDSS evaluation under both sufficient and scarce resource regimes, emphasizing restoration performance through priority-weighted service metrics and the role of feasibility through quantified operational constraint violations. The chapter then distills the dissertation's main contributions into a coherent set of methodological and evaluation advances, clarifies the primary limitations arising from the adopted assumptions (e.g., centralized control and snapshot-style simulation), and closes by outlining research directions that extend the work toward more deployment-realistic restoration, such as distributed control, real-time and hardware validation, joint switching-and-dispatch decision-making, and broader benchmark testing.

6.1 Conclusion

This study presented PDQN-CLR, a Parameterized Deep Q-Network framework for critical load restoration in post-disaster islanded microgrids, targeting the coupled problem of (i) maximizing priority-weighted load service under uncertainty in resource availability and (ii) maintaining feasibility with respect to DER and network operating constraints. The controller follows the parameterized (hybrid) action-space paradigm, in which discrete operational choices are paired with continuous control parameters, consistent with the established P-DQN formulation.

The proposed approach was evaluated on the IEEE 123-node distribution feeder implemented in OpenDSS under islanded operation. The adopted implementation partitions the feeder into five zones, each anchored by a grid-forming BESS, and models 17 critical loads organized into three priority tiers. This study explicitly modeled post-disaster uncertainty through a capacity-factor derating model and evaluated two regimes: Sufficient ($\alpha \in [0.45, 0.55]$) and Scarce ($\alpha \in [0.11, 0.14]$), with episode-wise randomization of fault scenario, capacity factor, and initial SOC over a 72-step (36-hour) horizon.

Across 150 evaluation episodes per regime, the headline results showed that under sufficient resources, PDQN-CLR attained a mean PCS = 0.94 versus PCS = 0.80 for the Greedy baseline (a 17.5% increase). More importantly, PDQN-CLR maintained CVS ≈ 0.009 in both regimes, whereas Greedy exhibited markedly higher violation scores (≈ 0.388 in sufficient and ≈ 0.667 in scarce). These correspond to reductions in constraint violations by factors of approximately 43 and 74, respectively. These findings reinforce the interpretation developed in Chapter Five: when scarcity compresses achievable restoration scores, feasibility metrics remain essential because quantity-only outcomes can obscure operational risks.

Tier-level results indicate that PDQN-CLR exhibits the intended priority-consistent trade-off under scarcity: P1 service increased to 100%, whereas P2 and P3 services decreased to 64.5% and 66.7%, respectively, yielding the reported prioritization asymmetry. Dispatch trajectory analysis further indicated resource-adaptive behavior. In the sufficient regime, PDQN-CLR sustained a high aggregate dispatch (≈ 250 kW) early in the episode and maintained PCS = 100% for approximately 21 steps (≈ 10.5 h) before

transitioning to a lower dispatch level. Under scarcity, the policy maintained a comparatively low and stable dispatch envelope, consistent with a capacity-limited regime.

From an implementation standpoint, the closed-loop simulation pipeline relies on OpenDSS snapshot-style power-flow evaluation at each decision step, which aligns with OpenDSS's scripting model for stepwise solves in Snapshot mode. Training is computationally tractable in the reported configuration: a full 3,000-episode (216,000-step) run completed in 0.89 h on an NVIDIA Tesla V100 (16 GB), with modest GPU memory usage (≈ 408 MiB) and sub-millisecond inference latency per action. Finally, the combination of adaptive Lagrangian penalties and pre-execution safety filtering is supported by the consistently low CVS observed in the evaluation.

6.2 Contributions

This work contributes to the field by proposing a unified learning-and-evaluation framework for post-disaster islanded restoration that reflects the mixed discrete–continuous nature of DER dispatch, incorporates explicit resource uncertainty, embeds critical-load priorities directly into the learning objective, and enforces feasibility through constraint-aware learning and safe execution.

First, the study formulates critical load restoration as a parameterized (hybrid) control problem, coupling discrete restoration actions with continuous DER setpoints, and implements the resulting controller within the P-DQN paradigm. This preserves the natural decision structure of restoration control while enabling continuous optimization over dispatch setpoints conditioned on discrete operational choices.

Second, an uncertainty-aware resource-availability model based on capacity-factor derating is introduced, and its performance is evaluated across sufficient and scarce regimes with episode-wise randomization of fault scenarios and initial SOC. This design moves beyond single-condition evaluations and provides evidence of policy behavior under both feasibility-permitting and capacity-limited conditions.

Third, this research embeds three-tier critical-load prioritization into both learning and reporting using PCS-driven reward shaping and tier service-rate analysis, demonstrating that the learned policy preserves the highest-priority tier under scarcity while allocating residual capacity to lower-priority tiers in a manner consistent with the objective.

Fourth, a DER-aware service evaluation that combines voltage admissibility with DER-support attribution for electrically distant buses is proposed, mitigating over-crediting that can arise in islanded studies when voltage-only indicators are influenced by the island reference configuration.

The following list is a summary of this work's contributions:

- Hybrid (parameterized) restoration control using discrete action templates paired with continuous DER setpoints.
- Uncertainty-aware evaluation via capacity-factor derating and episode randomization across sufficient and scarce regimes.
- Priority-consistent learning objective using the PCS/tier structure that yields P1 preservation under scarcity.
- DER-aware service crediting to reduce voltage-only bias in islanded OpenDSS evaluations.

6.3 Limitations

This section summarizes the principal limitations of the PDQN-CLR as implemented and evaluated in this study. These limitations motivate the extensions outlined in Section 6.4.

6.3.1 Centralized decision-making architecture.

The PDQN-CLR is implemented as a single agent that observes the full system state and issues coordinated setpoints for all five DERs. While this centralized design simplifies learning and ensures globally consistent decisions, it introduces scaling concerns as the system size grows. In the current implementation, the state is 209-dimensional, and the feature groups (bus voltages, DER operating points/limits, load-status features, and SOC) would expand with feeder size and monitoring granularity. Moreover, the centralized setting does not represent deployment conditions in which DER controllers operate with partial observability, limited bandwidth or non-negligible communication delays.

6.3.2 Steady-state (snapshot) simulation at a coarse time resolution.

The environment evaluates actions using OpenDSS Snapshot power-flow solves, executing one steady-state solution per decision step with a 30-minute dispatch interval. This modeling choice supports large-scale sampling for training; however, it abstracts fast electrical and control dynamics (e.g., transient voltage/frequency behavior during load pickup, inverter synchronization transients, and sub-second protection interactions). Consequently, the reported performance establishes feasibility and restoration behavior under steady-state operating points, rather than under electromagnetic-transient (EMT) dynamics or real-time timing constraints.

6.3.3 Dispatch-only control with a fixed post-fault topology.

The hybrid action space selects among 21 discrete categories and applies continuous DER P/Q parameters; however, it does not include network switching, reconfiguration or tie-line control. In the present setup, the faulted element is disabled at initialization, and the remaining network is solved under a fixed island reference configuration. Therefore, the agent cannot co-optimize restoration via combined switching and dispatch decisions (e.g., sectionalizing actions, microgrid formation, or topology-dependent energization sequences).

Also, training throughput is constrained by simulator-in-the-loop power flow. The training time is dominated by the per-step OpenDSS power-flow call executed on the CPU, yielding modest observed GPU utilization (≈3–20%) even when a V100 is available. This bottleneck limits the rapid scaling of larger feeders, longer horizons, or extensive hyperparameter sweeps without additional parallelization or acceleration mechanisms.

6.3.4 Limited scenarios and resource diversity.

The evaluation is reported for the IEEE 123-node feeder under three severe trunk-line fault scenarios, and the DER fleet is limited to five grid-forming BESS units under a capacity-derating model. The islanded configuration also relies on a single weak grid-forming voltage reference (Vsource at bus 44), and multi-island partitioning with inter-island coordination is beyond the scope of this study. These choices provide a controlled and challenging benchmark but constrain direct generalization to other feeder topologies, broader fault distributions, mixed DER portfolios (e.g., diesel, PV/wind intermittency), and multi-island restoration settings.

6.4 Future Work

Building on the limitations above, several directions can be explored to extend the PDQN-CLR toward large-scale and more deployment-realistic restoration control.

6.4.1 Distributed and multi-agent restoration control.

A natural extension is to replace the centralized controller with a distributed architecture in which each DER (or each zone) is controlled by a local agent with partial observations, and coordination is learned through multi-agent reinforcement learning. The centralized training with decentralized execution (CTDE) paradigm provides a practical training structure in which global information can be used during learning, whereas execution remains local and communication-aware.

Promising directions include zone-level agents (one agent per DER/zone), hierarchical coordination (system-level goals with local dispatch), and explicit modeling of communication constraints and dropouts. Recent restoration studies have demonstrated the value of MARL for combining DER dispatch and restoration decisions under changing network conditions.

6.4.2 Real-time and hardware validation.

To close the gap between steady-state simulations and deployment practices, future work should evaluate PDQN-CLR policies in real-time simulations and hardware-in-the-loop setups. Controller-HIL (CHIL) testing can assess timing, I/O handling, and delay/noise sensitivity, while power-HIL (PHIL) extends validation to power exchange effects and interface constraints. In addition, EMT-based studies are increasingly emphasized for inverter-dominated systems, providing a path to validate the stability and transient behavior during energization and mode transitions.

6.4.3 Joint switching and dispatch restoration.

Extending the hybrid action space to include switching and reconfiguration actions would enable the joint optimization of topology and DER injections, potentially improving restoration beyond dispatch-only control, particularly when reconfiguration can reduce the electrical distance to critical loads or isolate problematic segments. This direction aligns with published DRL restoration work that explicitly incorporates tie-switches or network reconfiguration control.

6.4.4 Computational scalability and acceleration.

Training efficiency can be improved by (i) running multiple simulator instances in parallel to increase the sample throughput and (ii) using carefully designed surrogate models to approximate expensive power-flow evaluations during the early training stages, with periodic correction using the full simulator to limit drift. Parallel environment engines and vectorized execution are standard methods for increasing the throughput of simulator-driven RL. In power-system contexts, surrogate modeling has been explored as a method to reduce the computational burden in optimization and simulation pipelines, and similar ideas can be adapted for restoration learning loops.

6.4.5 Broader benchmarks and operating regimes.

Finally, future work should expand the evaluation to additional feeders and operating conditions. For example, the IEEE 8500-node test feeder is commonly used in OpenDSS examples and provides a stress test for scalability and data management choices. Further extensions include richer fault sets (including lateral faults), mixed DER portfolios (PV/wind intermittency and fuel-limited generation), and explicit multi-island formation with inter-island coordination.

REFERENCES

[1] U.S. Department of Energy, "Economic Benefits of Increasing Electric Grid Resilience to Weather Outages," Executive Office of the President, Washington, DC, USA, Aug. 2013. [Online]. Available: https://www.energy.gov/sites/prod/files/2013/08/f2/Grid%20Resiliency%20Report_FINAL.pdf

[2] U.S. Census Bureau, "How Do Power Outages Affect Households?," American Housing Survey, Oct. 2024. [Online]. Available: https://www.census.gov/library/stories/2024/10/power-outages.html

[3] National Hurricane Center, "Hurricane Helene Tropical Cyclone Report," Mar. 2025. [Online]. Available: https://www.nhc.noaa.gov/data/tcr/AL092024_Helene.pdf

[4] M. Panteli and P. Mancarella, "The Grid: Stronger, Bigger, Smarter?: Presenting a Conceptual Framework of Power System Resilience," *IEEE Power Energy Mag.*, vol. 13, no. 3, pp. 58–66, May 2015, doi: 10.1109/MPE.2015.2397334.

[5] L. Fisher, "More than 70 ways to show resilience," *Nature*, vol. 518, no. 7537, pp. 35–35, Feb. 2015, doi: 10.1038/518035a.

[6] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[7] M. Bruneau *et al.*, "A Framework to Quantitatively Assess and Enhance the Seismic Resilience of Communities," *Earthq. Spectra*, vol. 19, no. 4, pp. 733–752, Nov. 2003, doi: 10.1193/1.1623497.

[8] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An Introduction to Deep Reinforcement Learning," 2018, doi: 10.48550/ARXIV.1811.12560.

[9] M. Gautam, "Deep Reinforcement Learning for Resilient Power and Energy Systems: Progress, Prospects, and Future Avenues," *Electricity*, vol. 4, no. 4, pp. 336–380, Dec. 2023, doi: 10.3390/electricity4040020.

[10] M. A. Igder and X. Liang, "Service Restoration Using Deep Reinforcement Learning and Dynamic Microgrid Formation in Distribution Networks," *IEEE Trans. Ind. Appl.*, vol. 59, no. 5, pp. 5453–5472, Sep. 2023, doi: 10.1109/TIA.2023.3287944.

[11] J. Zhao, F. Li, S. Mukherjee, and C. Sticht, "Deep Reinforcement Learning-Based Model-Free On-Line Dynamic Multi-Microgrid Formation to Enhance Resilience," *IEEE Trans. Smart Grid*, vol. 13, no. 4, pp. 2557–2567, Jul. 2022, doi: 10.1109/TSG.2022.3160387.

[12] M. M. Hosseini, L. Rodriguez-Garcia, and M. Parvania, "Hierarchical Combination of Deep Reinforcement Learning and Quadratic Programming for Distribution System Restoration," *IEEE Trans. Sustain. Energy*, vol. 14, no. 2, pp. 1088–1098, Apr. 2023, doi: 10.1109/TSTE.2023.3245090.

[13] Y. Du and D. Wu, “Deep Reinforcement Learning From Demonstrations to Assist Service Restoration in Islanded Microgrids,” *IEEE Trans. Sustain. Energy*, vol. 13, no. 2, pp. 1062–1072, Apr. 2022, doi: 10.1109/TSTE.2022.3148236.
[14] M. M. Hosseini and M. Parvania, “Resilient Operation of Distribution Grids Using Deep Reinforcement Learning,” *IEEE Trans. Ind. Inform.*, vol. 18, no. 3, pp. 2100–2109, Mar. 2022, doi: 10.1109/TII.2021.3086080.
[15] R. A. Jacob, S. Paul, S. Chowdhury, Y. R. Gel, and J. Zhang, “Real-time outage management in active distribution networks using reinforcement learning over graphs,” *Nat. Commun.*, vol. 15, no. 1, p. 4766, Jun. 2024, doi: 10.1038/s41467-024-49207-y.
[16] L. Vu, T. Vu, T. L. Vu, and A. Srivastava, “Multi-Agent Deep Reinforcement Learning for Distributed Load Restoration,” *IEEE Trans. Smart Grid*, vol. 15, no. 2, pp. 1749–1760, Mar. 2024, doi: 10.1109/TSG.2023.3310893.
[17] L. Vu, T. Vu, T.-L. Vu, and A. Srivastava, “Safe Exploration Reinforcement Learning for Load Restoration using Invalid Action Masking,” in *2023 IEEE Power & Energy Society General Meeting (PESGM)*, Orlando, FL, USA: IEEE, Jul. 2023, pp. 1–5. doi: 10.1109/PESGM52003.2023.10253213.
[18] H. Nie, Y. Chen, Y. Xia, S. Huang, and B. Liu, “Optimizing the Post-Disaster Control of Islanded Microgrid: A Multi-Agent Deep Reinforcement Learning Approach,” *IEEE Access*, vol. 8, pp. 153455–153469, 2020, doi: 10.1109/ACCESS.2020.3018142.
[19] M. Abdelmalak, H. Hosseinpour, E. Hotchkiss, and M. Ben-Idris, “Post-Disaster Generation Dispatching for Enhanced Resilience: A Multi-Agent Deep Deterministic Policy Gradient Learning Approach,” in *2022 North American Power Symposium (NAPS)*, Salt Lake City, UT, USA: IEEE, Oct. 2022, pp. 1–6. doi: 10.1109/NAPS56150.2022.10012133.
[20] B. Fan, X. Liu, G. Xiao, Y. Kang, D. Wang, and P. Wang, “Attention-Based Multiagent Graph Reinforcement Learning for Service Restoration,” *IEEE Trans. Artif. Intell.*, vol. 5, no. 5, pp. 2163–2178, May 2024, doi: 10.1109/TAI.2023.3314395.
[21] J. C. Bedoya, Y. Wang, and C.-C. Liu, “Distribution System Resilience Under Asynchronous Information Using Deep Reinforcement Learning,” *IEEE Trans. Power Syst.*, vol. 36, no. 5, pp. 4235–4245, Sep. 2021, doi: 10.1109/TPWRS.2021.3056543.
[22] Z. Ul Abdeen, X. Zhang, W. Gill, and M. Jin, “Enhancing Distribution System Resilience: A First-Order Meta-RL algorithm for Critical Load Restoration,” in *2024 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, Oslo, Norway: IEEE, Sep. 2024, pp. 129–134. doi: 10.1109/SmartGridComm60555.2024.10738096.
[23] B. Fan, X. Liu, G. Xiao, Y. Xu, X. Yang, and P. Wang, “A Memory-Based Graph Reinforcement Learning Method for Critical Load Restoration With Uncertainties of Distributed Energy Resource,” *IEEE Trans. Smart Grid*, vol. 16, no. 2, pp. 1706–1718, Mar. 2025, doi: 10.1109/TSG.2024.3482696.
[24] X. Zhang, A. T. Eseye, B. Knueven, W. Liu, M. Reynolds, and W. Jones, “Curriculum-Based Reinforcement Learning for Distribution System Critical Load

Restoration," *IEEE Trans. Power Syst.*, vol. 38, no. 5, pp. 4418–4431, Sep. 2023, doi: 10.1109/TPWRS.2022.3209919.
[25] Y. Liu, P. Liao, and Y. Wang, "Using Graph-Enhanced Deep Reinforcement Learning for Distribution Network Fault Recovery," *Machines*, vol. 13, no. 7, p. 543, Jun. 2025, doi: 10.3390/machines13070543.
[26] S. Huang, L. Ding, Y. Chen, Z. Yang, and T. Xiao, "Hybrid Policy Deep Reinforcement Learning-Based Robust Optimization for Emergency Dispatching of Multi-Park Integrated Energy Systems," 2023, *SSRN*. doi: 10.2139/ssrn.4568409.
[27] S. Kaloti and B. Chowdhury, "An Agent-Based Deep Reinforcement Learning Approach for Networked Microgrid Scheduling to Improve Resilience," in *2024 IEEE Industry Applications Society Annual Meeting (IAS)*, Phoenix, AZ, USA: IEEE, Oct. 2024, pp. 1–7. doi: 10.1109/IAS55788.2024.11023709.
[28] T. Zhang, M. Sun, D. Qiu, X. Zhang, G. Strbac, and C. Kang, "A Bayesian Deep Reinforcement Learning-Based Resilient Control for Multi-Energy Micro-Gird," *IEEE Trans. Power Syst.*, vol. 38, no. 6, pp. 5057–5072, Nov. 2023, doi: 10.1109/TPWRS.2023.3233992.
[29] Md. Kamruzzaman, J. Duan, D. Shi, and M. Benidris, "A Deep Reinforcement Learning-Based Multi-Agent Framework to Enhance Power System Resilience Using Shunt Resources," *IEEE Trans. Power Syst.*, vol. 36, no. 6, pp. 5525–5536, Nov. 2021, doi: 10.1109/TPWRS.2021.3078446.
[30] R. S. Sutton and A. Barto, *Reinforcement learning: an introduction*, Nachdruck. in Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2014.
[31] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 279–292, May 1992, doi: 10.1007/BF00992698.
[32] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol. 8, no. 3–4, pp. 293–321, May 1992, doi: 10.1007/BF00992699.
[33] M. Hausknecht and P. Stone, "Deep Reinforcement Learning in Parameterized Action Space," 2015, *arXiv*. doi: 10.48550/ARXIV.1511.04143.
[34] J. Xiong *et al.*, "Parametrized Deep Q-Networks Learning: Reinforcement Learning with Discrete-Continuous Hybrid Action Space," 2018, *arXiv*. doi: 10.48550/ARXIV.1810.06394.
[35] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward Constrained Policy Optimization," 2018, *arXiv*. doi: 10.48550/ARXIV.1805.11074.
[36] E. Altman, *Constrained Markov Decision Processes: Stochastic Modeling*, 1st ed. Boca Raton: Routledge, 2021. doi: 10.1201/9781315140223.
[37] A. Kushwaha, K. Ravish, P. Lamba, and P. Kumar, "A Survey of Safe Reinforcement Learning and Constrained MDPs: A Technical Survey on Single-Agent and Multi-Agent Safety," 2025, *arXiv*. doi: 10.48550/ARXIV.2505.17342.
[38] A. Stooke, J. Achiam, and P. Abbeel, "Responsive Safety in Reinforcement Learning by PID Lagrangian Methods," 2020, *arXiv*. doi: 10.48550/ARXIV.2007.03964.

[39] W. H. Kersting, “Radial distribution test feeders,” in *2001 IEEE Power Engineering Society Winter Meeting. Conference Proceedings (Cat. No.01CH37194)*, Columbus, OH, USA: IEEE, 2001, pp. 908–912. doi: 10.1109/PESW.2001.916993.
[40] Electric Power Research Institute (EPRI)., “OpenDSS Documentation,” 2024. [Online]. Available: https://opendss.epri.com/IntroductiontoOpenDSS.html
[41] D. Narang, R. Mahmud, M. Ingram, and A. Hoke, “An Overview of Issues Related to IEEE Std 1547-2018 Requirements Regarding Voltage and Reactive Power Control,” NREL/TP-5D00-77156, 1821113, MainId:26102, Sep. 2021. doi: 10.2172/1821113.
[42] Z. Chen, S. Cai, and A. P. S. Meliopoulos, “Robust Deep Reinforcement Learning for Volt-VAR Optimization in Active Distribution System under Uncertainty,” 2024, *arXiv*. doi: 10.48550/ARXIV.2409.18937.
[43] T. P. Lillicrap *et al.*, “Continuous control with deep reinforcement learning,” Jul. 05, 2019, *arXiv*: arXiv:1509.02971. doi: 10.48550/arXiv.1509.02971.
[44] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” Aug. 28, 2017, *arXiv*: arXiv:1707.06347. doi: 10.48550/arXiv.1707.06347.
[45] T. Haarnoja *et al.*, “Soft Actor-Critic Algorithms and Applications,” 2018, *arXiv*. doi: 10.48550/ARXIV.1812.05905.
[46] X. Zhang, B. Knueven, A. Zamzam, M. Reynolds, and W. Jones, “Primal-Dual Differentiable Programming for Distribution System Critical Load Restoration,” in *2023 IEEE Power & Energy Society General Meeting (PESGM)*, Orlando, FL, USA: IEEE, Jul. 2023, pp. 1–5. doi: 10.1109/PESGM52003.2023.10252454.
[47] J. Achiam, D. Held, A. Tamar, and P. Abbeel, “Constrained Policy Optimization,” May 30, 2017, *arXiv*: arXiv:1705.10528. doi: 10.48550/arXiv.1705.10528.
[48] S. Gu *et al.*, “A Review of Safe Reinforcement Learning: Methods, Theories, and Applications,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 11216–11235, Dec. 2024, doi: 10.1109/TPAMI.2024.3457538.
[49] H. van Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-learning,” 2015, *arXiv*. doi: 10.48550/ARXIV.1509.06461.
[50] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized Experience Replay,” 2015, *arXiv*. doi: 10.48550/ARXIV.1511.05952.
[51] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, “Deep Reinforcement Learning that Matters,” 2017, *arXiv*. doi: 10.48550/ARXIV.1709.06560.

Appendix A

Table A.1 Discrete action categories in the PDQN-CLR hybrid action space - Extended

k	Category	Template name	Involved DERs *I(k)*	Operational semantics
0	Single DER	Dispatch Bus 4	{4}	Update only the DER at Bus 4 from (P_1, Q_1); others retain prior setpoints
1	Single DER	Dispatch Bus 26	{26}	Update only the DER at Bus 26 from (P_2, Q_2); others retain prior setpoints
2	Single DER	Dispatch Bus 44	{44}	Update only the DER at Bus 44 from (P_3, Q_3); others retain prior setpoints
3	Single DER	Dispatch Bus 60	{60}	Update only the DER at Bus 60 from (P_4, Q_4); others retain prior setpoints
4	Single DER	Dispatch Bus 86	{86}	Update only the DER at Bus 86 from (P_5, Q_5); others retain prior setpoints
5	Zone dispatch	Zone 1 (primary at Bus 4)	{4}	Zone-centric template: updates the zone's primary DER only (same mask as k=0)
6	Zone dispatch	Zone 2 (primary at Bus 26)	{26}	Zone-centric template: updates the zone's primary DER only (same mask as k=1)
7	Zone dispatch	Zone 3 (primary at Bus 44)	{44}	Zone-centric template: updates the zone's primary DER only (same mask as k=2)
8	Zone dispatch	Zone 4 (primary at Bus 60)	{60}	Zone-centric template: updates the zone's primary DER only (same mask as k=3)
9	Zone dispatch	Zone 5 (primary at Bus 86)	{86}	Zone-centric template: updates the zone's primary DER only (same mask as k=4)
10	Priority restore	Priority-group A (P1-focused)	{4,26,44,60,86}	Fleet-wide template: all DER setpoints may be overwritten in one step
11	Priority restore	Priority-group B (non-P1-focused)	{4,26,44,60,86}	Fleet-wide template: all DER setpoints may be overwritten in one step
12	Multi-zone	Zones 1–2	{4,26}	Update the two involved zone DERs; others retain prior setpoints
13	Multi-zone	Zones 2–3	{26,44}	Update the two involved zone DERs; others retain prior setpoints

k	Category	Template name	Involved DERs *I(k)*	Operational semantics
14	Multi-zone	Zones 3–4	{44,60}	Update the two involved zone DERs; others retain prior setpoints
15	Multi-zone	Zones 4–5	{60,86}	Update the two involved zone DERs; others retain prior setpoints
16	Multi-zone	Zones 1–4	{4,60}	Update the two involved zone DERs; others retain prior setpoints
17	Multi-zone	Zones 2–5	{26,86}	Update the two involved zone DERs; others retain prior setpoints
18	Voltage support	System-wide voltage support (mode 1)	{4,26,44,60,86}	Fleet-wide Q injection *(Q > 0)* to raise low voltages (within limits).
19	Voltage support	System-wide voltage support (mode 2)	{4,26,44,60,86}	Fleet-wide Q absorption *(Q < 0)* to reduce high voltages (within limits).
20	No-op	Hold setpoints	∅	No DER is overwritten; all DERs keep (P,Q) from (t-1)