Energy and AI 19 (2025) 100457

Contents lists available at ScienceDirect

Energy and AI

journal homepage: www.elsevier.com/locate/egyai

RL-ADN: A high-performance Deep Reinforcement Learning environment for optimal Energy Storage Systems dispatch in active distribution networks☆

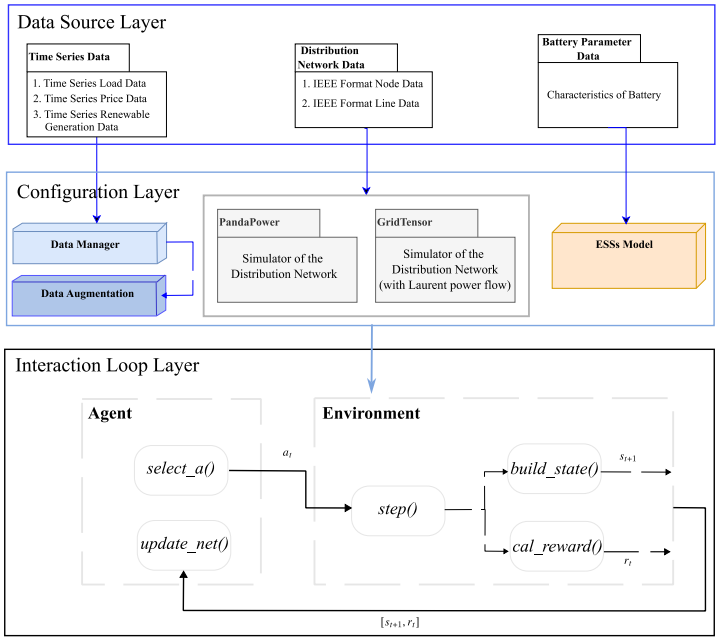
Hou Shengren [a], Gao Shuyi [a], Xia Weijie [a], Edgar Mauricio Salazar Duque [b], Peter Palensky [a], Pedro P. Vergara [a,*]

[a] Department of Electrical Sustainable Energy, Delft University of Technology, Mekelweg 4, Delft, 2628 CD, The Netherlands
[b] Energy Systems Systems Group, Eindhoven University of Technology, Eindhoven, 5612 AE, The Netherlands

HIGHLIGHTS

- Versatile Library: RL-ADN offers flexible DRL-based ESS dispatch in active distribution networks.
- Enhanced Training: GMC-based data augmentation improves training diversity and DRL performance.
- Efficient Solver: Tensor Power Flow solver reduces computation time tenfold, maintaining accuracy.

GRAPHICAL ABSTRACT



ARTICLE INFO

Dataset link: https://github.com/ShengrenHou/RL-ADN/tree/main/rl_adn/data_sources

Keywords:
Distribution networks
Battery dispatch

ABSTRACT

Deep Reinforcement Learning (DRL) presents a promising avenue for optimizing Energy Storage Systems (ESSs) dispatch in distribution networks. This paper introduces RL-ADN, an innovative open-source library specifically designed for solving the optimal ESSs dispatch in active distribution networks. RL-ADN offers unparalleled flexibility in modeling distribution networks, and ESSs, accommodating a wide range of research goals. A standout feature of RL-ADN is its data augmentation module, based on Gaussian Mixture Model and Copula

☆ This publication is part of the project ALIGN4Energy (with project number NWA.1389.20.251) of the research programme NWA ORC 2020 which is (partly) financed by the Dutch Research Council (NWO), The Netherlands This work is part of the DATALESs project (with project number 482.20.602) jointly financed by The Netherlands Organization for Scientific Research (NWO), and the National Natural Science Foundation of China (NSFC).

* Corresponding author.
E-mail address: P.P.VergaraBarrios@tudelft.nl (P.P. Vergara).

https://doi.org/10.1016/j.egyai.2024.100457
Received 14 August 2024; Received in revised form 11 November 2024; Accepted 30 November 2024
Available online 19 December 2024
2666-5468/© 2024 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Battery optimization
Machine learning
Voltage regulation

(GMC) functions, which elevates the performance ceiling of DRL agents, achieving an average performance improvement of 21.43%, 1.08%, 2.76%, by augmenting five-year, one-year and three-month data, respectively. Additionally, RL-ADN incorporates the Tensor Power Flow solver, significantly reducing the computational burden of power flow calculations during training without sacrificing accuracy, maintaining voltage magnitude with an average error not exceeding 0.0001%. The effectiveness of RL-ADN is demonstrated using distribution networks with size varying, showing marked performance improvements in the adaptability of DRL algorithms for ESS dispatch tasks. Furthermore, RL-ADN achieves a tenfold increase in computational efficiency during training, making it highly suitable for large-scale network applications. The library sets a new benchmark in DRL-based ESSs dispatch in distribution networks and it is poised to advance DRL applications in distribution network operations significantly. RL-ADN is available at: https://github.com/ShengrenHou/RL-ADN and https://github.com/distributionnetworksTUDelft/RL-ADN.

1. Introduction

1.1. Motivation

Energy Storage Systems (ESSs) play a pivotal role in modern distribution networks, offering enhanced flexibility essential for addressing uncertainties brought by Distributed Energy Resources (DERs) integration [1]. Optimizing ESS dispatch strategies is crucial for distribution system operators (DSOs) to fully harness this flexibility [2]. However, the dynamic and sequential nature of optimal operation decisions, responding to fluctuating prices and varying electricity demands, poses a significant challenge. Traditional model-based approaches often struggle with real-time decision-making due to their reliance on predefined forecasts or complex probability functions to manage uncertainties [3]. Deep Reinforcement Learning (DRL) emerges as a potent model-free solution for such fast-paced, sequential decision-making scenarios, with successful applications in diverse fields like game-playing [4], robotics control [5], industry control [6]. Applied to distribution energy systems, DRL transforms these operational challenges into a Markov Decision Process (MDP), exhibiting impressive results in various energy tasks [7–9]. DRL's strength lies in its adaptability and capability for real-time decision-making, trained in simulators and then applied to real-world scenarios. This necessitates robust and accurate simulation environments to prevent duplication and provide benchmark frameworks for the development of efficient DRL algorithms.

Therefore, we introduce RL-ADN, an open-source library specifically tailored for DRL-based optimal ESSs operation in distribution networks. It meets diverse research needs while providing customization options for research tasks, ensuring both flexibility and standardization.

1.2. Related work

The RL field has grown significantly, thanks in part to open-source universal simulation environments and benchmark frameworks, like GYM for game-playing [4]. However, this trend is less pronounced in energy system research groups. The absence of such resources hampers the development and integration of DRL algorithms in energy system operation areas. Table 1 offers a comparative analysis of functionalities in open-sourced energy system environments. Many existing environments address specific challenges but are often too tailored for broader application [10,11]. For instance, a microgrid environment is developed to test the performance of DRL algorithms in [11]. The task of formulated MDP is to minimize the power unbalance and operational cost by dispatching distributed generators and ESSs. In the research [12], a distribution network environment is open-sourced to facilitate solving active voltage control problems based on multi-agent RL algorithms. AndesGYM [13] developed an environment for frequency control problems in power systems, which leverages the modeling capability of ADNES and Gym environment. The task is set to minimize the deviations of the frequency value in a given time scope. Consequently, these environments do not lend themselves easily to customization or alterations essential for different or broader research objectives. This specificity leads to fragmentation in the research community, as studies operate in isolation without a standardized benchmark or a universally adaptable toolset.

CityLearn [14] provides an environment for simulating DRL algorithms in charge of operating building energy systems, in either a centralized (single-agent) or decentralized (multi-agent) way. Focusing on exploring the dynamics inside the building, it ignored the grid-level dynamic. GridLearn [15] is further developed to investigate mitigating over-voltages in the distribution network level by demand response in the buildings. Both two packages simplified the original MDP tasks, by discrete continuous decisions into discrete actions and ignoring the power flow calculation in the distribution networks. PowerGridWorld [16] is a framework for researchers to customize multi-agent environments of power networks, which could integrate existing RL libraries like RLLIb and OPEN-AI BASELINES. PowerGridWorld could work in two ways to implement the multi-agent feature: centralized training and distributional execution, distributional training, and execution. In the environment, OPENDSS is used as an interface to execute the power network operation. Gird2OP [17] is developed to support training an intelligent agent to run a transmission network and has served as a benchmark environment for a series of L2RPN competitions. Grid2OP provided the flexibility for grid modifications, observations, and actions. However, both PowerGridWorld and Grid2OP necessitate extensive power flow calculations during offline training, typically a bottleneck in DRL training, since RL agents need to explore the environments to converge, requiring a large amount of interaction. The mentioned electricity network environments are mainly built based on standard iterative methods, i.e. Newton–Raphson method, which is time-consuming, rendering them unsuitable for integration with DRL algorithms training.

GYM-ANM [18] is an open-source environment for solving operation problems in distribution networks, with the primary purpose of using RL algorithms to reduce energy loss (including generation curtailment storage, and transmission losses) under the operation violation constraints. GYM-ANM provides flexibility for customizing energy components, research tasks, network topology, etc. Specifically, it uses a customized simplified power flow simulator to encapsulate the dynamics of a distributional network, which can accelerate the training speed of RL agents significantly. However, the limitations of GYM-ANM are also obvious, as the implemented power flow calculation algorithm cannot precisely track the dynamic of physical distribution networks, impeding the transition from simulation to reality for the trained RL agents. Therefore, an advanced power flow calculation algorithm remains a significant imperative to avoid being hindered by the extensive computational demands as well as to reflect the dynamics of physical distribution networks accurately.

Moreover, the key to leveraging DRL for optimal dispatch strategies lies in training with diverse historical data, particularly in environments with uncertain renewable generation, load consumption, and price profiles. The broader the training scenarios, the higher the DRL agents' performance ceiling [11]. However, collecting diverse data for specific distribution networks remains challenging, limiting the practical integration of DRL algorithms.

Table 1
Summary of literature in environments of distribution network operation. The content of the table strictly aligns with the novelty we include: power flow integration, data augmentation, benchmark optimality, and flexibility assessment.

Work	Research task	Power flow integration	Data augmentation	Flexibility and customization capabilities
[11]	Optimal energy system scheduling	×	×	×
[12]	Voltage regulation	✓	×	×
CityLearn [14]	Building Energy Management	×	×	✓
GridLearn [15]	Building Energy Management	×	×	✓
PowerGridWorld [16]	Power Network Operation	✓	×	✓
Grid2OP [17]	Transmission Network Configuration	✓	×	✓
GYM-ANM [18]	Distribution Network Operation	✓	×	✓
[3]	Microgrid operation	×	×	×
[19]	EV energy management	×	×	×
[20]	Microgrid Control	✓	×	×
[21]	Microgrid operation	✓	×	✓
[22]	Economic dispatch	×	×	×
[23]	Power system emergency control	✓	×	✓
[24]	Voltage Control	✓	×	×
RL-ADN	Optimal ESSs dispatch in distribution network	✓	✓	✓

1.3. Contributions

This paper presents RL-ADN, an open-source library for DRL-based optimal ESSs dispatch in active distribution networks. RL-ADN accommodates a wide range of research objectives (i.e., different optimization objectives functions such as congestion management and optimal dispatch) while offering unprecedented customization capabilities. This flexibility extends to the modeling of distribution network topologies and the integration of various types of ESSs, thereby allowing for the creation of tailored MDPs. RL-ADN incorporates a novel data augmentation module using a Gaussian Mixture Models-Copula (GMC) approach, enhancing the diversity of training scenarios and thereby the performance of DRL algorithms. Additionally, it introduces the Tensor Power Flow solver, drastically reducing computation time for power flow calculations tenfold, without sacrificing accuracy [25,26]. RL-ADN also provides four state-of-the-art (SOTA) DRL algorithms and a model-based approach with perfect forecasts as a standard baseline for comparison. In summary, RL-ADN sets a new standard in DRL-based ESS dispatch with its innovative features, flexibility, and efficiency. The proposed environment paves the way for more effective and accurate DRL applications in energy distribution networks, representing a significant advancement in the field.

2. Background

2.1. Optimal ESS dispatch tasks in distribution networks

ESSs dispatch tasks are inherently sequential decision-making problems. The aim is to minimize operational costs while adhering to constraints that ensure the safe and efficient operation of the distribution network. Such constraints might include maintaining specific voltage magnitude and current levels, state of charge (SOC) operation constraints, etc. This involves responding to market prices, network conditions, and renewable stochastic generation. The ESSs dispatch problem is typically cast as optimization problems with a general mathematical optimization formulation defined by (1)–(3):

Minimize:

$$f(x) \quad \text{where } x \text{ is the decision variable.} \tag{1}$$

Subject to:

$$g(x) < y \qquad \text{(Grid-level constraints)} \tag{2}$$

$$b(x) < z \qquad \text{(Energy storage system constraints)} \tag{3}$$

The objective function $f(x)$ varies based on different tasks, ranging from minimizing operation cost based on dynamic pricing to regulating voltage magnitude or integrating multiple goals [27]. The effective dispatch of ESSs is crucial, considering the uncertainties in renewable generation, load consumption, and price fluctuations. The constraints are categorized into grid-level (2) and ESS-level (3) based on the specific requirements of the tasks. Some tasks may prioritize network reliability and incorporate more stringent constraints on voltage magnitude and current levels, while others may focus solely on profit maximization. This flexibility in formulation allows for a wide array of approaches, each tailored to meet the specific needs and priorities of different energy optimization tasks. Detailed mathematical formulation for a template task can be found in Appendix A.

2.2. MDP formulation and reinforcement learning

In RL-ADN, these sequential decision-making problems can be reformulated as a MDP, defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ represents the action space, $\mathcal{P}$ is the state transition probability function, $\mathcal{R}$ signifies the reward function, and γ stands for the discount factor.

A policy, $\pi(a_t|s_t)$, determines the selection of action a_t for a given state s_t. The agent's objective is to ascertain a policy that maximizes the expected discounted cumulative return, represented as $J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\mathcal{T}} \gamma^t r_t\right]$, in which $\mathcal{T}$ is the length of the control horizon.

The formulated MDP possesses a continuous action space, making it unsuitable for direct solutions using value-based DRL algorithms [28]. Policy-based DRL algorithms are often employed to address continuous action spaces, as they directly tackle such continuous action domain problems. The architectures of state-of-the-art (SOTA) policy-based DRL algorithms such as DDPG [29], TD3 [30], SAC [31], and PPO [32] are depicted in Fig. 1.

- **DDPG and TD3:** Both are deterministic algorithms that maintain a policy for action sampling and Q-networks, $Q_\theta(s_t, a_t)$, to guide policy network updates. Specifically, TD3, as an enhancement of DDPG, incorporates dual Q-networks and employs delayed updates, mitigating the Q-network's overestimation bias inherent in DDPG.
- **PPO:** As an on-policy algorithm, PPO addresses policy optimization challenges in RL. PPO curtails extensive policy updates by adopting a clipped objective function, ensuring minimal deviation of the new policy from the previous one. A value function $V_\phi(s)$ is leveraged to guide the policy iteration. This mechanism circumvents the necessity of learning rate adjustments and achieves superior sample efficiency compared to conventional policy gradient techniques [32].
- **SAC:** SAC is an off-policy actor-critic framework that integrates the maximum entropy reinforcement learning paradigm. By supplementing the typical reward with an entropy component, SAC promotes exploration, thereby achieving a harmonious balance between exploration and exploitation. This algorithm utilizes a

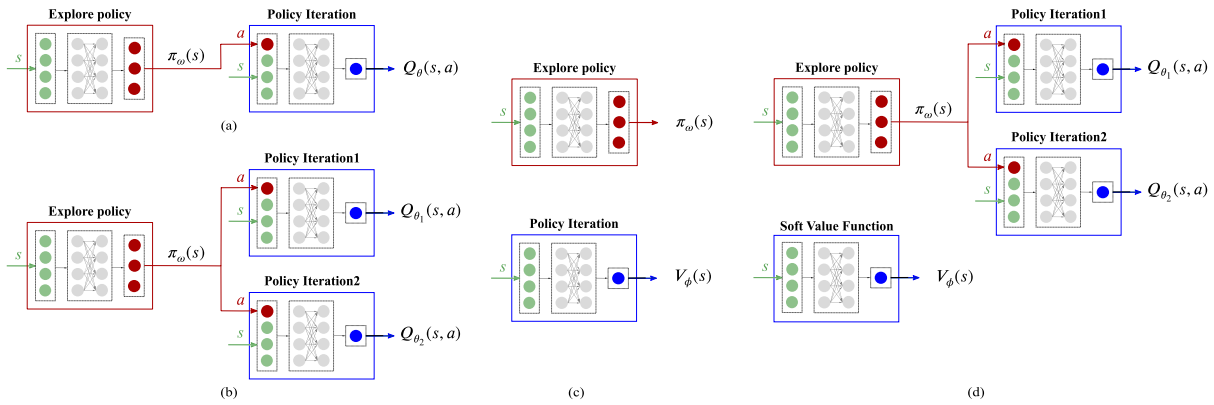


Fig. 1. Architecture of policy-based DRL algorithms. (a) Deep Deterministic Policy Gradient (DDPG), (b) Twin Delayed DDPG (TD3), (c) Proximal Policy Optimization (PPO), (d) Soft Actor-Critic (SAC).

soft value function, dual Q-functions, and a policy network. With iterative updates, SAC strives to formulate a stochastic policy that is both optimal and exploratory, ensuring robustness and efficiency across diverse tasks.

Building on the policy gradient theorem, both the policy, $\pi(a_t|s_t)$, and its associated critic networks, $Q_\theta(s_t, a_t)$ or $V_\phi(s)$, can be updated. It is worth noting that the update methods can vary depending on the specific algorithm. A comprehensive discussion of these algorithms is available in [33].

By interacting with the artificial environment, the DRL agent seeks to define the optimal ESSs dispatch in active distribution networks. The two-phase approach, offline training followed by online deployment, equips the agent to address the stochastic nature of optimal ESSs dispatch tasks. In the offline training phase, the DRL agent gleans insights from the interaction and executes self-learning, refining its decision-making. During the subsequent online deployment, it leverages these insights to navigate complexities, ensuring more robust and adaptive solutions. The environment's partially observable nature, often due to communication constraints, necessitates meticulous state selection from the full observation set. Overly complex states will decrease the signal-to-noise ratio, while overly simplistic states could overlook essential dynamics. Both scenarios can undermine the learning efficacy and policy performance. To provide flexibility in designing state spaces, RL-ADN facilitates the easy customization of state spaces, a topic further explored in the subsequent sections.

3. RL-ADN framework

3.1. Overview

The architecture of the RL-ADN environment, depicted in Fig. 2, consists of three layers: Data Source, Configuration, and Interaction Loop. Primary data feed into Configuration Layer to build DRL environments, integrating components like Data Manager, Distribution Network Simulator, and ESSs Models. These components are integrated into the environment within the Interaction Loop, while a DRL algorithm, chosen to control the agent, is initialized simultaneously.[1] Then, the DRL agent interacts with the environment in search of the optimal policy. The proposed RL-ADN framework's versatility allows for modeling highly tailored tasks, with modifications to components yielding unique MDPs for distinct ESSs dispatch tasks.

[1] State-of-the-art policy-based algorithms such as DDPG, SAC, TD3, and PPO are incorporated into the framework.

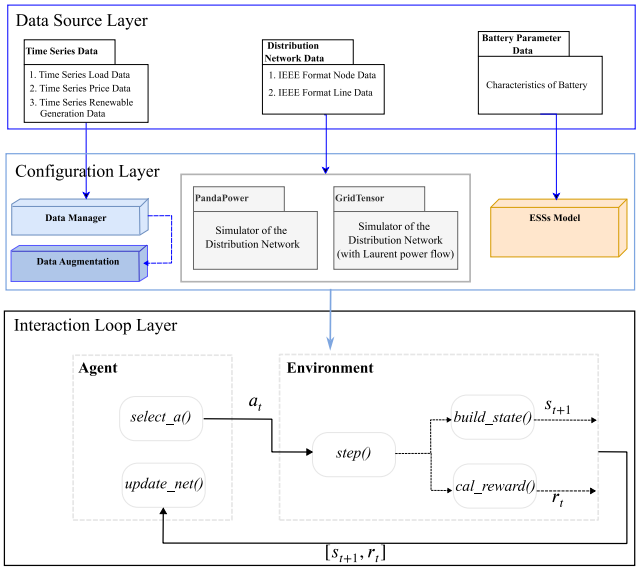


Fig. 2. Framework of the RL-ADN package. Configuration data for the distribution network and the ESSs are selected from data sources. Subsequently, corresponding time-series data undergo preprocessing. Through Configuration Layer, the environment is constituted of the distribution network, ESSs, and data manager.

3.2. Data source layer

The Data Source Layer provides primary data for building the framework and training the DRL agent. Data are categorized into time-series data, distribution network data, and ESSs parameter data. Time-series data include load profiles, price profiles, and renewable generation profiles in a standard format. These data are processed by the Data Manager for training or can be selected for further augmentation. Distribution network data comprises node and line data, with nodes specifying slack and PQ bus locations and lines detailing topology and characteristics like resistance and reactance, which are stored in CSV format. This data is crucial for building the distribution network simulator. ESSs parameter data, detailing capacity, charge/discharge limitation, and degeneration costs, are used to construct the ESSs model. The framework includes standard 25, 34, 69, and 123 node distribution network data, along with corresponding time-series data and ESSs data from previous research [10]. Users can use this data for training or customize their own model following the provided standard format.

3.3. Configuration layer

3.3.1. Data manager

The Data Manager plays a crucial role in managing time-series data, such as active and reactive power demand ($p^D_{i,t}$, $q^D_{i,t}$), electricity price (ρ_t), and renewable power generation ($p^R_{i,t}$, $q^R_{i,t}$) for specific epochs ($\mathcal{T}, t \in \mathcal{T}$). Previous research approaches to data management have been case-specific and labor-intensive, adding complexity and potential data quality issues. RL-ADN adopts a streamlined approach, standardizing various data preprocessing tasks, and ensuring data integrity and efficient handling. The workflow of the Data Manager is detailed in Appendix B.1.

3.3.2. Data augmentation

In RL-ADN, Data Augmentation module plays a pivotal role in enhancing the robustness and generalizability of the trained policy by artificially expanding the diversity of the historical time-series data. With data augmentation, RL-ADN exposes the model to a broader set of scenarios, promoting adaptability and performance in varied and unforeseen situations.

The Data Augmentation module is designed to generate synthetic time-series data, capturing the stochastic nature of load in the power system and reflecting realistic operational conditions. The Data Augmentation module interacts with the Data Manager to retrieve the necessary preprocessed data and then applies its augmentation algorithms to produce an augmented dataset. The output is a synthetic yet realistic dataset that reflects the variability and unpredictability inherent in distribution network systems. This enriched dataset is crucial for training RL agents, providing them with a diverse range of scenarios to learn from and ultimately resulting in a more adaptable and robust decision-making policy. The workflow of Data Augmentation module is described in Appendix B.2.

3.3.3. Distribution network simulator

For a distribution network, node-set $\mathcal{N}$ and the line set $\mathcal{L}$ define the topology. Each of the node $i \in \mathcal{N}$ and lines $l_{i,j} \in \mathcal{L}$ specify its attributes. A specific subset $\mathcal{B}, \mathcal{B} \subset \mathcal{N}$ describes ESSs connected to the distribution network nodes. Importantly, the number of ESSs delineates the resulting state space $\mathcal{S}$ and action space $\mathcal{A}$.

The main function of the Distribution Network Simulator is to calculate power flow when a new scenario is fed into the environment, performing as the main part of the state transition function for the formulated MDP task. Based on the provided distribution network configuration data, we offer two modules, `PandaPower` and `GridTensor`, to create the Distribution Network Simulator. `PandaPower` provides the traditional iterative methods while `GridTensor` [26] integrates a fast Tensor Power Flow for calculating the distribution network state presented by the voltage magnitudes, currents and power flowing in the lines.

3.4. Interaction loop layer

For each time step t in an episode, the agent obtains the current state s_t and determines an action a_t to be executed in the environment. Once a_t is received, the environment will execute `step` function to execute power flow and update the status of ESSs and the distribution network, which is counted as the consequence of the action at the current time step t. Then, based on these resultant observations, the reward r_t is calculated by the designed reward calculation block. Next, the Data Manager in the environment samples external time-series data of the next time step $t+1$, including demand, renewable energy generation, and price, emulating the stochastic fluctuations of the environment. These external variables are combined with updated internal observations, performing as the resultant transition of the environment.

Users can freely design the `build-state` block, facilitating an in-depth exploration of how different states influence the performance of algorithms on various tasks. In a similar vein, the `cal-reward` block can be tailored according to different optimal tasks. For the convenience of our users, our framework provides a default state pattern and reward calculation.

3.5. MDP design

3.5.1. State space design

State space design is vital as it directly impacts the efficacy of the agent's learning process. The chosen state space $\mathcal{S}$ should be concise yet descriptive enough to facilitate effective policy learning.

In the RL-ADN framework, the environment collects a comprehensive range of measurements at each timestep t. Using all these measurements to represent the state s_t in the MDP is plausible but fraught with challenges. Such an approach might not be practical in real-world distribution networks due to potential data unavailability. Moreover, by including all measurements, the state space could become noise-prone, making state exploration more intricate and possibly hindering agent performance.

Thus, feature engineering is pivotal in designing state s_t. The RL-ADN framework offers the flexibility to tailor state space. The `get-obs` block fetches available measurements, while the `build-state` block lets users customize states. Generally, the state s_t encompasses both endogenous and exogenous features. Exogenous features capture external dynamics, like uncertainties in renewable energies, consumption, and pricing, within an episode. Meanwhile, endogenous features track internal dynamics governed by distribution network rules and energy component behaviors, e.g., power flow and ESS's SOC update rules. Moreover, some ancillary information, such as the current time-step in a trajectory, has proven crucial in MDP state representation [27].

3.5.2. Action space design

Focusing the optimal ESS dispatch tasks, the action a_t at time t is denoted as $a_t = [p^B_{m,t}|_{m\in\mathcal{B}}]$, symbolizing the charging or discharging directives for the m_{th} ESS connected to node m in the distribution network.

3.5.3. Transition function

In a MDP, the transition function encapsulates the dynamics that govern the system's progression from one state to another. The transition mechanism is bifurcated into two essential components. The first is endogenous distribution network and energy component dynamics. These are calculated based on physical laws, i.e., power flow calculation, SOC update rules, rooted in the network's topology, the variations in active and reactive power at different nodes, and the parameters of ESSs models. The second is exogenous variable evolution, which involves modeling the temporal fluctuations in renewable energy generation, market prices, and load demand, leveraging daily historical data. The transition probability function $\mathcal{P}$ is mathematically represented as:

$$p(S_{t+1}, R_t | S_t, A_t) = \Pr\left\{S_{t+1} = s_{t+1}, R_t = r_t \mid S_t = s_t, A_t = a_t\right\}. \tag{4}$$

Traditionally, constructing a precise mathematical representation of $\mathcal{P}$ has been challenging due to the inherent complexities and uncertainties in both endogenous and exogenous variables. Reinforcement Learning (RL) offers a way around this by learning the ambiguous model through interaction.[2]

[2] Model-free RL algorithms obviate the need for explicit knowledge of $\mathcal{P}$, enabling the agent to learn optimal policies through interaction with the environment.

3.5.4. Reward function

The reward function serves as a critical component for guiding the agent's learning process. The environment offers a reward signal r_t to the agent, quantifying the quality of each action taken. The design of this reward function is inherently tied to the specific objectives of the task at hand[3]. Our framework incorporates a `cal-reward` block that allows researchers to easily customize the reward signal for various optimal ESS dispatch challenges.

3.6. Data augmentation model

The RL-ADN framework incorporates Gaussian mixture models (GMM) and Copula functions for data augmentation [34,35]. The GMM is a probabilistic model that assumes data originates from a blend of multiple Gaussian distributions, each characterized by unique means and covariances. This model can adeptly capture the complex and multi-modal nature of time series data in distribution networks, which often exhibit intricate patterns due to fluctuating load demands and renewable energy generation. Complementing the GMM, Copula functions are utilized to encapsulate the time-correlation structure between multiple time-step data in a defined period, independent of their marginal distributions. This dual approach ensures a comprehensive and realistic augmentation of time-series data in distribution network operations. In our framework, three augmentation methods are provided: GMM, t-Copula, and Gaussian Copula [36].

The integration of GMM and Copula functions (GMC) in the RL-ADN framework marks a significant advancement in creating robust and reliable environments for training reinforcement learning agents. This approach adeptly handles the complexities and uncertainties inherent in power distribution networks, enhancing the quality of training data and the effectiveness of the resulting policies.

3.7. Tensor power flow

Conventional power flow calculations often rely on iterative methods like the Newton–Raphson algorithm. This becomes a computational bottleneck, especially in the context of training DRL agents, which requires numerous evaluations of power flow. In the proposed framework, we address the computational bottleneck associated with traditional power flow calculations by incorporating a Tensor Power Flow algorithm [26]. This efficiency approach is achieved by linearizing the power flow equations using a Laurent series expansion, simplifying the nodal current calculations in the distribution network [25]. By doing so, we facilitate frequent power flow evaluations necessary for training RL agents without the computational burden.

The Tensor Power Flow method considers constant power and impedance loads, integrating the ZIP load model directly into the power flow analysis. This approach allows for the inclusion of various types of loads and renewable energy sources without the need for iterative approximation methods typically used in traditional power flow analysis. As a result, our algorithm achieves rapid convergence and permits a more streamlined and scalable RL training process. The elimination of iterative computation not only expedites the power flow assessment but also enhances the RL agent's ability to quickly adapt and learn, thereby improving the overall efficiency and effectiveness of the framework.

4. Benchmark scheme and experiments

4.1. Optimal ESSs dispatch task and MDPs

RL-ADN framework introduces a foundational optimal ESSs dispatch case while the mathematical formulation of the case is shown in Appendix A. This default case aims to minimize the operational costs for DSOs while ensuring compliance with the distribution network and ESSs operation constraints. The template case offers researchers and practitioners a springboard, enabling them to design bespoke benchmarks tailored to unique ESSs dispatch challenges.

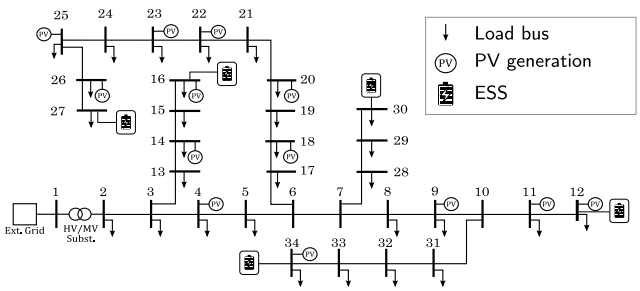


Fig. 3. Modified IEEE-34 Node bus test system with distributed PV generation and EESs. The ESSs are placed at the end of each feeder to increase the number of voltage magnitude issues experienced.

In the provided case, a modified 34-node IEEE test distribution network is leveraged to build the Distribution Network Simulator, as illustrated in Fig. 3. Strategic placement of the ESSs on nodes 12, 16, 27, and 34, which have over- and under-voltage issues. The objective remains to minimize the operational cost while upholding voltage magnitude constraints. Consequently, the state and reward functions are constructed as below: the state s_t is described as $s_t = [P^N_{m,t}|_{m\in\mathcal{N}}, \rho_t, SOC^B_{m,t}|_{m\in\mathcal{B}}]$, incorporating both endogenous and exogenous features. The design of $\mathcal{A}$ adheres to the optimal goal and multiple constraints:

- **Charge and Discharge Bounds:** ESSs have inherent physical limitations. The action a_t is confined within a range, considering these physical constraints.
- **State-of-Charge (SOC) Dependency:** Actions must respect the current SOC of each ESS. The 'step' function ensures this by adjusting the charge/discharge commands based on SOC levels.
- **Voltage Magnitude Regulation:** ESS actions should maintain voltage within predefined limits. Direct enforcement is infeasible; hence, we employ soft constraints via penalty rewards for voltage violations.

Thus, the reward function is defined as the combination of energy arbitrage profits and the penalty of the voltage magnitude violations in the distribution network. Mathematically, this is expressed as:

$$r_t = \rho_t \left[\sum_{m\in\mathcal{N}} \left(P^B_{m,t}\right)\right] \Delta t - \sigma \left[\sum_{m\in\mathcal{B}} C_{m,t}(V_{m,t})\right], \tag{5}$$

where $C_{m,t}$ is constraint violation functions [3]:

$$C_{m,t} = \min\left\{0, \left(\frac{\overline{V}-\underline{V}}{2} - |V_0 - V_{m,t}|\right)\right\}, \forall m \in \mathcal{B}. \tag{6}$$

where σ is a trade-off parameter between energy arbitrage and voltage stability.

4.2. Bench-marking approach

To assess performance, we formulate the optimal ESS dispatch problem as a model-based optimization problem, with ESS dispatch decisions as the primary variables. Historical data — including renewable generation, load consumption, and market prices — are treated as perfect forecasts and inputted into the optimization model. Solving this model yields a globally optimal solution, serving as a benchmark for evaluating DRL-derived strategies. Following previous research [10], we can assess the efficiency of DRL algorithms by defining performance

[3] The default reward functions are presented in Section 4.1.

Table 2
Summary - Parameters for DRL algorithms and the MDP.

PPO Alg.	$\gamma = 0.995$ Optimizer = Adam Learning rate=$6e-4$ Batch size = 4096 GAE parameter(λ) = 0.99
DDPG, TD3 Alg.	$\gamma = 0.995$ Optimizer = Adam Learning rate=$6e-4$ Batch size = 512 Replay buffer size = $4e5$
SAC Alg.	$\gamma = 0.995$ Optimizer = Adam Learning rate=$6e-4$ Batch size = 512 Entropy=auto
Reward	$\sigma = 400$
ESSs	$\overline{p}^B = 50$ kW, $\underline{p}^B = -50$ kW, $\overline{SOC}^B = 0.8$, $\underline{SOC}^B = 0.2$,
Voltage limit	$\overline{v} = 1.05$, $\underline{v} = 0.95$

bound:

$$\text{Performance Bound} = \frac{C_{DRL} - C_{opt}}{C_{opt}} \tag{7}$$

Where C_{DRL} is the operational cost of the dispatch strategy derived from DRL agents, while C_{opt} is that derived from the global optimal solution. The closer the DRL decisions align with this benchmark, the higher the efficacy of the RL agents. We incorporate SOTA DRL algorithms capable of handling continuous action spaces, such as DDPG, PPO, SAC, and TD3, as our benchmark DRL algorithms.

Following prior research [3], our simulation dataset comprises electricity market prices from the Netherlands, augmented with consumption and PV generation data at a 15-minute resolution. Hyperparameter settings for the utilized DRL algorithms are detailed in Table 2. We compare the performance of these DRL algorithms against global optimal solutions obtained by formulating Nonlinear Programming (NLP) problems, solved using the Pyomo package [37].

5. Results

5.1. Performance of DRL algorithms on template optimal dispatch task

Fig. 4 displays the average total reward, operational cost, and the number of voltage magnitude violations during the training process for DDPG, SAC, TD3, and PPO algorithms. Results shown in Fig. 4 are obtained as an average of over five random seeds. The average total reward increases rapidly during the training, while simultaneously, the number of voltage magnitude violations decreases. This is a typical training trajectory of DRL algorithms solving optimal dispatch formulated MDP tasks, especially for those using penalty as a reward. At the beginning of the training process, the DNN's parameters are randomly initialized, and as a consequence, the actions defined usually are random discharge/charge decisions, causing a high number of voltage magnitude violations, thus introducing a huge magnitude penalty term in reward (5). Such a reward acts as an indicator to guide updating the DNN's parameters, resulting in higher quality actions, primarily learning to reduce voltage magnitude violations. Then, after reducing the violations, DRL algorithms learn to improve the actions toward increasing and minimizing the operational costs. All these DRL algorithms converged at around 1000 episodes. The total reward of these algorithms converged at 7.5 ± 0.02. Notice that even converged, the operational cost shown in Fig. 4(b) will not remain the same because the different daily load and price profiles are sampled during the training. After the last training episode, the penalty voltage magnitude

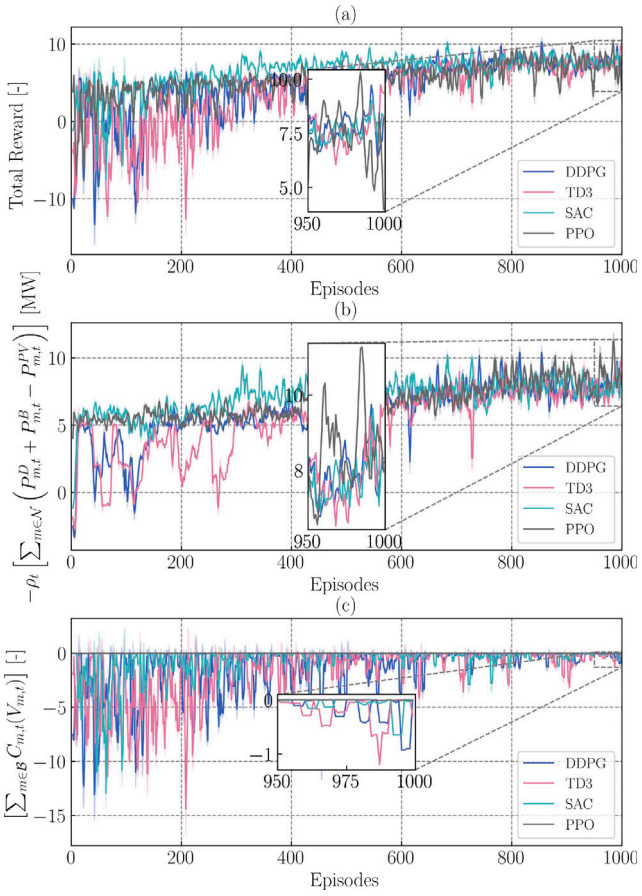


Fig. 4. (a) Average total reward as in (5). (b) Operational cost or first term of reward in (5). (c) Cumulative penalty for voltage magnitude violations or second term of reward in (5), all during training.

violation penalty for these DRL algorithms was reduced to a value of no more than 1 as is shown in Fig. 4(c). This result shows that DRL algorithms can effectively learn from interactions, reducing the number of voltage magnitude violations while minimizing the operational costs by learning to dispatch the ESSs correctly.

Fig. 5 shows the dispatch decisions and SOC changes of the ESS, connected to node 16 in a typical daily operation. These decisions are defined by DDPG, TD3, PPO, and SAC, as well as the global optimality benchmark solution provided by solving the NLP formulation considering the perfect forecast. Decisions provided by all DRL algorithms all responded to the dynamic prices during the day. On this day, PPO and SAC perform better than DDPG and TD3. Between 1:00-5:00, when the electricity price is low, PPO and SAC dispatch the ESS in charging mode, which is similar to the decisions from the NLP solver. However, DDPG and TD3 fail to learn to act efficiently with the low prices in these timeslots. During the afternoon, all DRL algorithms charge ESSs between low-price slots while discharging between high-price time slots (see Fig. 5(b) and (c)). However, Both DRL algorithms fail to capture the price fluctuations perfectly, compared to the decisions from NLP with full observation of the future. For instance, DDPG performs best among all DRL algorithms between 14:00 and 20:00 but fails to capture the price fluctuations well in the morning. PPO generally performs well during the whole day's operation but defines conservative decisions from 6:00 to 14:00.

Compared to the solution provided by NLP, all DRL algorithms converge to a local optimum after training in the current historical dataset. This performance can be caused by the limited scenarios in the training dataset, which hinder the implication of DRL algorithms in the

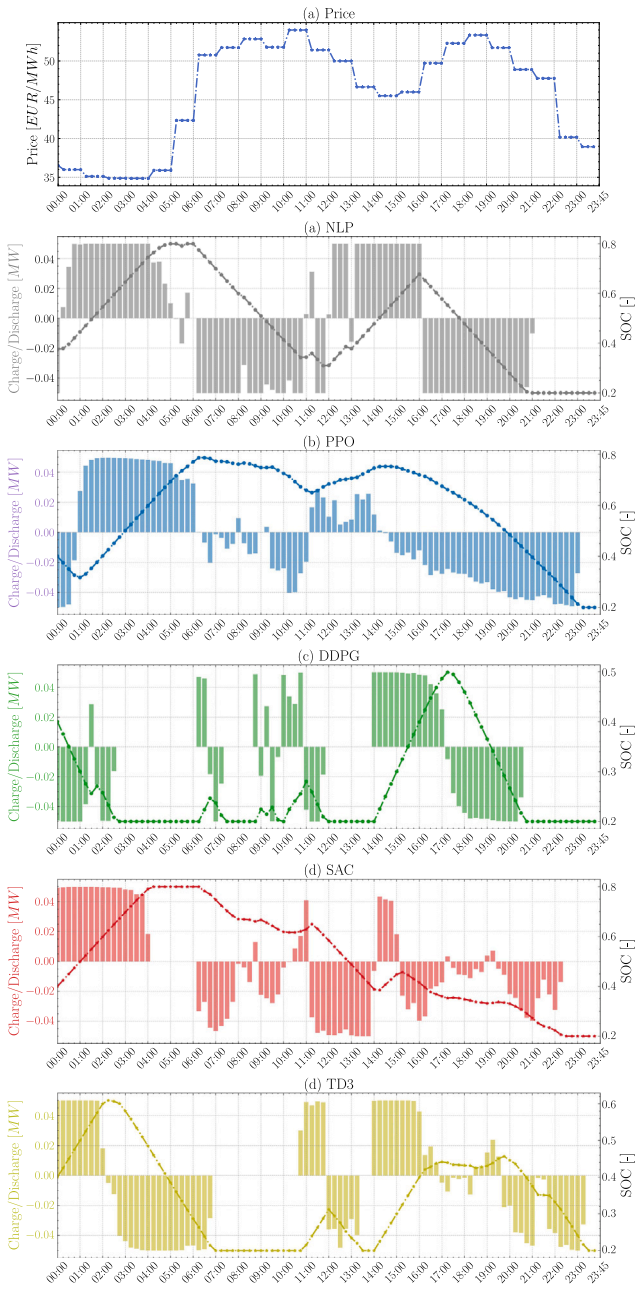


Fig. 5. Dispatch decisions obtained by DRL algorithms and NLP for the ESS connected to node 16.

realistic optimal dispatch operation. In the next section, we show how the performance of DRL algorithms is significantly influenced by using the data augmentation model incorporated in the RL-ADN framework.

5.2. Impacts of data augmentation on performance of DRL algorithms

The original data and results generated by the GMC model are depicted in Fig. 6. The GMC model captured the original patterns of peaks and valleys and diverse scenarios between different nodes in the testing distribution network. For instance, in the original data, the daily consumption profiles at around noon are diverse, where some nodes equipped with ESSs have negative load consumption (discharged), while others show peaks of daily consumption. The developed GMC model replicates such diversity.

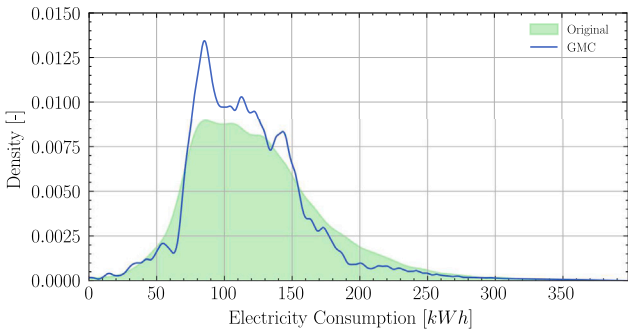


Fig. 6. Distribution of the original and generated data.

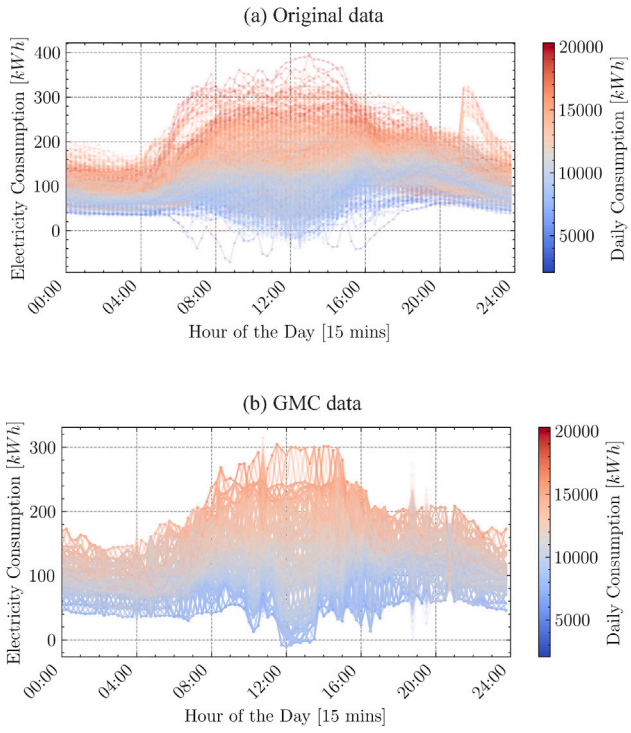


Fig. 7. Original and GMC generated load profiles. The color of the profiles corresponds to the sum of daily consumption.

Fig. 7 shows the original and generated data distribution shape. Both original and generated data have a long tail distribution. The shape of the GMC augmentation model's distribution matches the original data's shape. Therefore, the generated data profiles can enhance the scenario diversity without losing the original distribution and time correlation in the original dataset.

Table 3 presents the average reward, voltage magnitude violation penalty, and performance bounds for DRL algorithms on a separate 30-day test dataset. These algorithms, trained on primary datasets of 1 month, 3 months, and 1 year, were further augmented to 1 year and 5 years to examine the effects of data augmentation within the RL-ADN framework. Consistency in training parameters was maintained across 1000 episodes, and the results include 95% confidence intervals.

Initially, the performance of DRL algorithms using 1-month data was suboptimal. For example, the PPO algorithm's highest performance bound was below 70% (69.1%). However, post-augmentation, there was a significant improvement: PPO's performance increased to 84.0% and 85.9% with 1-year and 5-year data augmentation, respectively. When trained on 3-month primary data, DRL algorithms demonstrated good performance, which was further enhanced with data augmentation. For instance, TD3 improved from 80.6% to 82.2% with 1-year

Table 3
Mean and 95% confidence bounds for reward, violation penalty and performance bound.

Primary dataset	Augmented dataset	Reward [-]	Violation penlaty [-]	Performance bound [%]
One month	No augmentation	DDPG (3.40±0.86)	DDPG (0.0±0.0)	DDPG (51.1±6.7)
		PPO (5.91±0.91)	PPO (−0.002±0.001)	PPO (69.1±4.8)
		SAC (4.825±0.62)	SAC (0.0±0.0)	SAC (62.5±4.1)
		TD3 (3.49±0.88)	TD3 (0.0±0.0)	TD3 (52.4±7.0)
	augment 1 year	DDPG (9.55±0.88)	DDPG (−1.05±-0.77)	DDPG (82.8±1.1)
		PPO (11.625±0.92)	PPO (−0.039±-0.01)	PPO (84.0±1.0)
		SAC (9.95±0.63)	SAC (−0.25±-0.01)	SAC (83.4±0.5)
		TD3 (10.565±0.91)	TD3 (−0.09±-0.01)	TD3 (83.9±0.9)
	augment 5 year	DDPG (7.37±0.92)	DDPG (−0.32±-0.22)	DDPG (76.35±4.31)
		PPO (12.59±0.88)	(PPO-2.10±-0.69)	PPO (85.9±1.07)
		SAC (8.25±0.69)	SAC (−0.18±-0.09)	SAC (79.58±1.93)
		TD3 (8.02±0.91)	TD3 (−0.96±-0.41)	TD3 (78.82±2.67)
Three Month	No augmentation	DDPG (8.54±0.99)	DDPG (0.0±0.0)	DDPG (80.4±2.3)
		PPO (6.73±0.97)	PPO (0.0±0.0)	PPO (73.5±4.2)
		SAC (6.92±0.72)	SAC (0.0±0.0)	SAC (74.3±3.1)
		TD3 (8.60±0.92)	TD3 (0.0±0.0)	TD3 (80.6±2.1)
	augment 1 year	DDPG (9.38±0.99)	DDPG (0.0±0.0)	DDPG (82.5±1.4)
		PPO (9.68±0.94)	PPO (0.0±0.0)	PPO (83.0±1.0)
		SAC (7.78±0.55)	SAC (0.0±0.0)	SAC (78.0±1.9)
		TD3 (9.24±0.92)	TD3 (0.0±0.0)	TD3 (82.2±1.4)
	augment 5 year	DDPG (9.24±0.89)	DDPG (0.0±0.0)	DDPG (82.19±1.4)
		PPO (8.72±0.97)	PPO (0.0±0.0)	PPO (81.01±3.1)
		SAC (6.02±0.71)	SAC (0.0±0.0)	SAC (69.71±3.75)
		TD3 (8.45±0.95)	TD3 (0.0±0.0)	TD3 (80.20±3.32)
One year	No augmentation	DDPG (7.061±0.93)	DDPG (−0.01±0.0)	DDPG (75.0±3.7)
		PPO (8.173±1.02)	PPO (0.0±0.0)	PPO (79.3±2.8)
		SAC (7.302±0.84)	SAC (0.0±0.0)	SAC (76.1±3.2)
		TD3 (7.325±1.03)	TD3 (0.0±0.0)	TD3 (76.2±3.8)
	augment 5 year	DDPG (7.58±0.79)	DDPG (0.0±0.0)	DDPG (77.20±2.76)
		PPO (8.91±0.87)	PPO (0.0±0.0)	PPO (81.44±1.71)
		SAC (8.47±0.86)	SAC (0.0±0.0)	SAC (80.26±2.12)
		TD3 (7.99±0.99)	TD3 (0.0±0.0)	TD3 (78.72±2.90)

augmentation. Similarly, algorithms trained on one-year primary data showed good performance with minimal test set violations, and augmentation yielded incremental performance gains, as seen with PPO's increase from 79.3% to 81.44%. These results underscore the significance of data augmentation in enhancing the adaptation of DRL algorithms to varied market conditions, particularly for algorithms like DDPG and TD3. In scenarios with limited original datasets, the data augmentation module in the RL-ADN framework can substantially raise the performance ceiling of DRL algorithms.

However, a concerning observation was the increase in voltage magnitude violations in the 1-month data set trained algorithms post-augmentation, particularly notable with the 5-year augmentation. This could be attributed to the augmented data increasing scenario diversity but not altering the data distribution, as illustrated in Fig. 7. In such cases, while DRL algorithms perform better within the existing data distribution, they may incur violations in extreme scenarios not encountered during training. Notably, algorithms trained on more diverse datasets (three-month and one-year) exhibited better control over voltage violations. This is likely because these datasets encompassed the extreme scenarios present in the test sets. Yet, when comparing performance, algorithms trained on the one-year dataset displayed a lower performance ceiling than those trained on the three-month dataset. This suggests that while the one-year data provides a more diverse training environment, leading to potentially better generalization, it also presents a slower learning curve due to its complexity.

Generally, results indicate that in scenarios with limited original datasets, the data augmentation module in the RL-ADN framework can substantially raise the performance ceiling of DRL algorithms. Moreover, the distribution of data and the diversity of scenarios significantly impact the performance of DRL algorithms. Scenario diversity raises the performance ceiling, while data distribution affects the training difficulty and performance in extreme scenarios. While augmentation improves overall performance, it introduces complexities like increased violation penalties, especially when the primary dataset has a limited data distribution.

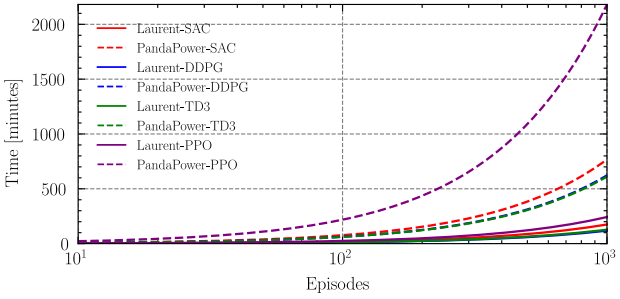


Fig. 8. Training time for DRL algorithms with Tensor Power Flow and Panda Power. The 34-node distribution network is used as a benchmark.

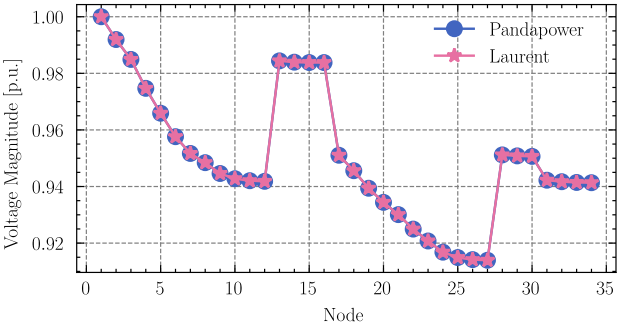


Fig. 9. Voltage magnitude calculated by Tensor Power Flow and Panda Power. The 34-node distribution network is used as a benchmark.

5.3. Enhancement of computation efficiency

The performance comparison between Tensor Power Flow and PandaPower power flow was conducted across multiple scale distribution

Table 4
Average calculation time comparison between Tensor Power Flow and Panda Power power flow for different scale distribution networks.

Distribution networks	Tensor power flow		Panda power	
	Power flow [ms]	Env. steps [ms]	Power flow [ms]	Env. steps [ms]
25 Nodes	**0.59**	2.81	28.08	30.30
34 Nodes	**0.61**	2.830	29.42	30.502
69 Nodes	**0.88**	2.99	28.72	31.46
123 Nodes	**0.97**	3.43	37.22	38.51

networks with node sizes: 25, 34, 69, and 123. The summarized results in Table 4 indicate a distinct computational advantage for the Tensor Power Flow method over PandaPower. First, Tensor Power Flow consistently maintained its efficiency, taking less than 1 ms across all node sizes. This starkly contrasts with PandaPower, which requires approximately 28 to 37 ms. In the smallest node size (25 nodes), Tensor Power Flow is about 47 times faster than PandaPower when solving one-time power flow. As the node size grows to 123, the efficiency margin increases, with Tensor Power Flow being nearly 38 times faster.

For executing one-time environment iteration, Tensor Power Flow's time ranges from 2.8 to 3.4 ms, while PandaPower's duration extends from 30 to 38 ms. This indicates that, on average, Tensor Power Flow is about ten times faster than PandaPower in processing environment steps, regardless of the node size. Overall, the Tensor Power Flow displays a significant computational edge, particularly as the node size expands. This relative efficiency is pivotal in training DRL algorithms in large-scale distribution networks. The ability of the Tensor Power Flow to consistently outpace PandaPower across different node sizes underscores its scalability, making it a more versatile choice for varied applications.

The comparison between Tensor Power Flow and Panda Power flow algorithms across different DRL algorithms showcases significant time differences in training for the same number of episodes as shown in Fig. 8. A clear trend emerges from the data: the Tensor Power Flow consistently outperforms PandaPower in terms of computational efficiency. For the SAC algorithm, the Tensor Power Flow is approximately 4.4 times faster than the PandaPower flow. Similarly, for DDPG, the Tensor Power Flow method shows a speedup of around 5.2 times. The TD3 algorithm with the Tensor Power Flow technology is about 4.8 times faster. The most pronounced difference is observed in the PPO algorithm, where Tensor Power Flow is significantly faster, clocking at approximately 9.1 times the speed of PandaPower. PPO requires 2200 min for training, making it the least efficient in this scenario. This is because PPO is an off-policy algorithm that cannot fully use the past experiences in the replay buffer, resulting in the lowest data efficiency and training speed. On the other hand, DDPG emerges as the fastest, closely followed by TD3 and then SAC.

In conclusion, the Tensor Power Flow demonstrates a clear computational advantage across all tested algorithms. While the choice of algorithm also affects the training time, with PPO consistently taking the longest, the underlying power flow technology plays a crucial role in determining the overall efficiency. These findings can guide researchers and practitioners in making informed decisions when selecting the most efficient combination of power flow technology and reinforcement learning algorithm.

Fig. 9 displays the voltage magnitude results of a 34-node distribution network from Tensor Power Flow and PandaPower flow, respectively. The voltage magnitude results from both algorithms remain almost the same magnitude, with an average error of no more than 0.0001%. Such high precision from Tensor Power Flow can track the real voltage dynamics accurately. Moreover, integrating Tensor Power Flow with the developed environment can significantly save the time cost for a large magnitude power flow iteration during the training. Thus, our framework can accelerate the training speed of DRL algorithms without losing simulation precision.

6. Discussion

The RL-ADN environment offers enhanced flexibility and customization, surpassing existing frameworks like CityLearn and GYM-ANM, which exhibit limited adaptability in modeling complex distribution networks. CityLearn focuses on building-level energy management, simplifying grid-level dynamics, while GYM-ANM lacks precision for complex network modeling. These limitations restrict the effectiveness of RL agents in real-world deployment. In contrast, RL-ADN provides extensive customization options, allowing researchers to model complex network topologies, integrate diverse ESSs, and design tailored MDPs. This flexibility helps bridge the sim-to-real gap, as demonstrated by RL-ADN's ability to adapt to complex pricing and load conditions more effectively than traditional frameworks.

The proposed RL-ADN environment includes a data augmentation module based on a GMC approach, significantly enhancing training scenario diversity and improving DRL performance. Unlike other frameworks that converge to local optima due to limited data, RL-ADN enables agents to learn from a broader range of scenarios, resulting in more effective policies. This addresses a key limitation of frameworks like PowerGridWorld and Grid2OP, where limited data diversity restricts real-world applicability.

Existing environments, such as those using PandaPower, face high computational demands, reducing efficiency for DRL training. PandaPower-based solutions can take tens of milliseconds for each power flow iteration, becoming a bottleneck during training. RL-ADN integrates the Tensor Power Flow solver, which achieves a tenfold increase in speed compared to PandaPower, greatly accelerating DRL training without sacrificing accuracy. Fig. 9 shows that Tensor Power Flow results closely match those results from PandaPower, ensuring realistic and efficient training.

While RL-ADN demonstrates significant advancements, there are limitations to its current implementation. One key challenge is the gap between simulation and reality, as building an accurate distribution network simulator is difficult [38]. This can lead to discrepancies when deploying RL agents trained in simulation to real-world environments. Another limitation is the potential difficulty in extending RL-ADN to integrated energy systems, such as transportation or hydrogen networks [39]. These systems introduce additional layers of complexity and require further development to handle their unique dynamics and computational requirements. Future work will focus on addressing these limitations by enhancing the accuracy of the distribution network simulations and extending the framework to integrated energy systems, including transportation and hydrogen, to improve the applicability and robustness of RL-ADN in diverse real-world conditions.

Overall, RL-ADN sets a new benchmark in applying DRL to dispatch ESSs tasks in distribution networks, offering a comprehensive solution that addresses the limitations of existing environments.

7. Conclusion

This paper unveiled RL-ADN, an open-sourced library tailored for designing and implementing DRL environments for optimal ESS dispatch challenges in modern distribution networks. We highlighted the potential of advanced DRL algorithms, showcasing their capacity to yield near-optimal decisions. The first significant innovation of our

approach is the seamless integration of the Tensor Power Flow, which offers unparalleled computational advantages over traditional methods, achieving more than tenfold faster. Another innovation is that RL-ADN integrates the Gaussian mixture model and Copula functions to augment the training dataset, thus further improving the performance ceiling for DRL algorithms. We believe RL-ADN presents a unique and extensive platform for future DRL research in energy systems. This research underscores the potential of a modular, customizable, and efficient RL environment to address the complexities of the energy landscape. We anticipate that RL-ADN will inspire a new wave of studies in the energy domain, leveraging its adaptability and precision.

CRediT authorship contribution statement

Hou Shengren: Writing – original draft, Validation, Software, Methodology, Conceptualization. **Gao Shuyi:** Writing – review & editing. **Xia Weijie:** Writing – review & editing. **Edgar Mauricio Salazar Duque:** Writing – review & editing. **Peter Palensky:** Funding acquisition. **Pedro P. Vergara:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Mathematical formulation of optimal ESS dispatch tasks

The template energy arbitrage task can be formulated by using the nonlinear programming (NLP) formulation given by (8)–(14). The objective function in (1) is extended to (8), aiming to minimize the total operational cost over the time horizon $\mathcal{T}$, comprising the cost of importing power from the main grid. The operational cost ρ_t at time slot t is settled according to the balancing market prices ρ_t in EUR/MWh.

$$\min_{P^B_{m,t},\forall m\in\mathcal{B},\forall t\in\mathcal{T}} \left\{ \sum_{t\in\mathcal{T}} \left[\rho_t \sum_{m\in\mathcal{N}} \left(P^D_{m,t} + P^B_{m,t} - P^{PV}_{m,t} \right) \Delta t \right] \right\}. \tag{8}$$

Subject to:

$$\sum_{nm\in\mathcal{L}} P_{nm,t} - \sum_{mn\in\mathcal{L}} (P_{mn,t} + R_{mn} I^2_{mn,t}) + P^B_{m,t} + P^{PV}_{m,t} + P^S_{m,t} = P^D_{m,t} \quad \forall m \in \mathcal{N}, \forall t \in \mathcal{T} \tag{9}$$

$$\sum_{nm\in\mathcal{L}} Q_{nm,t} - \sum_{mn\in\mathcal{L}} (Q_{mn,t} + X_{mn} I^2_{mn,t}) + Q^S_{m,t} = Q^D_{m,t} \quad \forall m \in \mathcal{N}, \forall t \in \mathcal{T} \tag{10}$$

$$V^2_{m,t} - V^2_{n,t} = 2(R_{mn}P_{mn,t} + X_{mn}Q_{mn,t}) + (R^2_{mn} + X^2_{mn}) I^2_{mn,t} \quad \forall m,n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{11}$$

$$V^2_{m,t} I^2_{mn,t} = P^2_{mn,t} + Q^2_{mn,t} \forall m,n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{12}$$

$$SOC^B_{m,t} = SOC^B_{m,t-1} + \eta^B_m P^B_{m,t} \Delta t / \overline{E}^B_m \ \forall m \in \mathcal{B}, \forall t \in \mathcal{T} \tag{13}$$

$$\underline{SOC}^B_m \le SOC^B_{m,t} \le \overline{SOC}^B_m \quad \forall m \in \mathcal{B}, \forall t \in \mathcal{T} \tag{14}$$

$$\underline{P}^B_m \le P^B_{m,t} \le \overline{P}^B_m \quad \forall m \in \mathcal{B}, \forall t \in \mathcal{T} \tag{15}$$

$$\underline{V}^2 \le V^2_{m,t} \le \overline{V}^2 \quad \forall m \in \mathcal{N}, \forall t \in \mathcal{T} \tag{16}$$

$$0 \le I^2_{mn,t} \le \overline{I}^2_{mn} \quad \forall mn \in \mathcal{L}, \forall t \in \mathcal{T} \tag{17}$$

$$P^S_{m,t} = Q^S_{m,t} = 0 \quad \forall m \in \mathcal{N}\backslash\{1\}, \forall t \in \mathcal{T} \tag{18}$$

The grid level constraints are modeled using the power flow formulation shown in (9)–(12) in terms of the active $P_{mn,t}$ power, reactive power $Q_{mn,t}$ and current magnitude $I_{mn,t}$ of lines, and the voltage magnitude $V_{m,t}$ of nodes. (16) and (17) enforce the voltage magnitude and line current limits, respectively, while (18) enforces that only one node is connected to the substation. The energy storage system constraints are modeled by (13)–(15). Eq. (13) models the dynamics of the ESSs' SOC on the set $\mathcal{B}$, while (14) enforces the SOC limits. Hereafter, it is assumed that the ESS $m \in \mathcal{B}$ is connected to node m, thus, $\mathcal{B} \subseteq \mathcal{N}$. Finally, (15) enforces the ESSs discharge/charge operation limits, Notice that to solve the above-presented sequential decision problem, all long-term operational data (e.g., expected PV generation and consumption) must be collected to properly define the EESs' dispatch decisions, while the power flow formulation must also be considered to enforce the voltage and current magnitude limits.

Appendix B. Workflows for modules in RL-ADN

B.1. Data manager workflow

`GeneralPowerDataManager` modular, is a unified data manager. Designed for automation, this class standardizes various data preprocessing tasks as follows:

- Loads time-indexed data directly from standard CSV files.
- Classifies columns pertaining to active and reactive power, renewable energy generation, and electricity pricing autonomously.
- Clean and check the data, filling in missing values, ensuring data continuity and integrity.
- Segregates the dataset into distinct training and test sets based on temporal delineation.
- Offers utility methods, such as `select-timeslot-data` and `select-day-data`, enabling precise data extraction tailored to the RL training needs.

When the `GeneralPowerDataManager` class is initialized, it undergoes a series of operations: it verifies the data's integrity, replaces any NaN values, and partitions the dataset into training and testing parts as required. These preliminary tasks ensure that data quality is maintained and provide ease of access and utilization for subsequent RL training processes.

B.2. Data augmentation workflow

The augmentation process involves several sophisticated statistical techniques, outlined as follows:

- The `ActivePowerDataManager` class, a subclass of the `GeneralPowerDataManager`, preprocesses the input data, fills missing values through interpolation, and restructures the data into an appropriate format for augmentation.
- A Gaussian Mixture Model (GMM) is fitted to the marginal distribution of historical active power data for each node and time step, capturing the underlying distribution of power consumption.
- The Bayesian Information Criterion (BIC) is employed to select the optimal number of components for each GMM, ensuring that the model complexity is balanced against the goodness of fit.
- A Copula-based approach is then applied, which models the dependency structure between different nodes and time steps, allowing for the generation of synthetic data points that maintain the correlation observed in historical data.
- The `augment_data` method leverages the GMM and Copula to produce new data samples, which are then transformed from the probabilistic space back to the power data scale.

The `TimeSeriesDataAugmentor` modular interacts with the data manager to retrieve the necessary preprocessed data, and then applies its augmentation algorithms to produce an augmented dataset. The output is a synthetic yet realistic dataset that reflects the variability and unpredictability inherent in power systems. This enriched dataset is crucial for training RL agents, providing them with a diverse range of scenarios to learn from and ultimately resulting in a more adaptable and robust decision-making policy.

Upon completion of the augmentation process, the synthetic data is saved to a CSV file, facilitating easy integration into the training pipeline. This automated and sophisticated data augmentation procedure enhances the RL-ADN framework's capability to train more effective and resilient RL agents for the distribution network ESSs operations.

Data availability

The data is available within the package https://github.com/ShengrenHou/RL-ADN/tree/main/rl_adn/data_sources.

References

[1] Specht JM, Madlener R. Deep reinforcement learning for the optimized operation of large amounts of distributed renewable energy assets. Energy AI 2023;11:100215.

[2] Vergara PP, López JC, Rider MJ, da Silva LCP. Optimal operation of unbalanced three-phase islanded droop-based microgrids. IEEE Trans Smart Grid 2019;10(1):928–40.

[3] Shengren H, Vergara PP, Salazar Duque EM, Palensky P. Optimal energy system scheduling using a constraint-aware reinforcement learning algorithm. Int J Electr Power Energy Syst 2023;152:109230.

[4] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, et al. Openai gym. 2016, arXiv preprint arXiv:1606.01540.

[5] Kaufmann E, Bauersfeld L, Loquercio A, Müller M, Koltun V, Scaramuzza D. Champion-level drone racing using deep reinforcement learning. Nature 2023;620(7976):982–7.

[6] Degrave J, Felici F, Buchli J, Neunert M, Tracey B, Carpanese F, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. Nature 2022;602(7897):414–9.

[7] Gallego F, Martín C, Díaz M, Garrido D. Maintaining flexibility in smart grid consumption through deep learning and deep reinforcement learning. Energy AI 2023;13:100241.

[8] Karagiannopoulos S, Aristidou P, Hug G, Botterud A. Decentralized control in active distribution grids via supervised and reinforcement learning. Energy AI 2024;16:100342.

[9] Vergara PP, Salazar M, Giraldo JS, Palensky P. Optimal dispatch of PV inverters in unbalanced distribution systems using reinforcement learning. Int J Elec Power Energy Syst 2022;136:107628.

[10] Hou S, Salazar EM, Palensky P, Chen Q, Vergara PP. A mix-integer programming based deep reinforcement learning framework for optimal dispatch of energy storage system in distribution networks. J Mod Power Syst Clean Energy 2024;1–13.

[11] Shengren H, Salazar EM, Vergara PP, Palensky P. Performance comparison of deep RL algorithms for energy systems optimal scheduling. In: 2022 IEEE PES innovative smart grid technologies conference Europe. IEEE; 2022, p. 1–6.

[12] Wang J, Xu W, Gu Y, Song W, Green TC. Multi-agent reinforcement learning for active voltage control on power distribution networks. Adv Neural Inf Process Syst 2021;34:3271–84.

[13] Cui H, Zhang Y. Andes_gym: A versatile environment for deep reinforcement learning in power systems. In: 2022 IEEE power & energy society general meeting. IEEE; 2022, p. 01–5.

[14] Vázquez-Canteli JR, Dey S, Henze G, Nagy Z. CityLearn: Standardizing research in multi-agent reinforcement learning for demand response and urban energy management. 2020, arXiv preprint arXiv:2012.10504.

[15] Pigott A, Crozier C, Baker K, Nagy Z. GridLearn: Multiagent reinforcement learning for grid-aware building energy management. Electr Power Syst Res 2022;213:108521.

[16] Biagioni D, Zhang X, Wald D, Vaidhynathan D, Chintala R, King J, et al. Powergridworld: A framework for multi-agent reinforcement learning in power systems. In: Proceedings of the thirteenth ACM international conference on future energy systems. 2022, p. 565–70.

[17] Donnot B. Grid2op- A testbed platform to model sequential decision making in power systems. 2020, GitHub repository, GitHub, https://GitHub.com/rte-france/grid2op.

[18] Henry R, Ernst D. Gym-ANM: Reinforcement learning environments for active network management tasks in electricity distribution systems. Energy AI 2021;5:100092.

[19] Xu J, Li Z, Gao L, Ma J, Liu Q, Zhao Y. A comparative study of deep reinforcement learning-based transferable energy management strategies for hybrid electric vehicles. In: 2022 IEEE intelligent vehicles symposium. 2022, p. 470–7.

[20] Bode H, Heid S, Weber D, Hüllermeier E, Wallscheid O. Towards a scalable and flexible simulation and testing environment toolbox for intelligent microgrid control. 2020, arXiv preprint arXiv:2005.04869.

[21] Lerousseau M. Design and implementation of an environment for Learning to Run a Power Network (L2RPN). 2021, arXiv preprint arXiv:2104.04080.

[22] de Mars P, O'Sullivan A. Applying reinforcement learning and tree search to the unit commitment problem. Appl Energy 2021;302:117519.

[23] Huang Q, Huang R, Hao W, Tan J, Fan R, Huang Z. Adaptive power system emergency control using deep reinforcement learning. IEEE Trans Smart Grid 2020;11(2):1171–82.

[24] Cui W, Li J, Zhang B. Decentralized safe reinforcement learning for voltage control. 2021, arXiv preprint arXiv:2110.01126.

[25] Giraldo JS, Montoya OD, Vergara PP, Milano F. A fixed-point current injection power flow for electric distribution systems using laurent series. Electr Power Syst Res 2022;211:108326.

[26] Duque EMS, Giraldo JS, Vergara PP, Nguyen PH, Slootweg HJ. Tensor power flow formulations for multidimensional analyses in distribution systems. Int J Electr Power Energy Syst 2024;162:110275.

[27] Salazar Duque EM, Giraldo JS, Vergara PP, Nguyen P, van der Molen A, Slootweg H. Community energy storage operation via reinforcement learning with eligibility traces. Electr Power Syst Res 2022;212:108515.

[28] Sutton RS, Barto AG. Reinforcement learning: An introduction. MIT Press; 2018.

[29] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, et al. Continuous control with deep reinforcement learning. 2015, arXiv preprint arXiv:1509.02971.

[30] Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods. In: International conference on machine learning. PMLR; 2018, p. 1587–96.

[31] Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning. PMLR; 2018, p. 1861–70.

[32] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. 2017, arXiv preprint arXiv:1707.06347.

[33] Chen X, Qu G, Tang Y, Low S, Li N. Reinforcement learning for decision-making and control in power systems: Tutorial, review, and vision. 2021, arXiv preprint arXiv:2102.01168.

[34] Bernards R, Morren J, Slootweg H. Statistical modelling of load profiles incorporating correlations using copula. In: 2017 IEEE PES innovative smart grid technologies conference Europe. IEEE; 2017, p. 1–6.

[35] Duque EMS, Vergara PP, Nguyen PH, van der Molen A, Slootweg JG. Conditional multivariate elliptical copulas to model residential load profiles from smart meter data. IEEE Trans Smart Grid 2021;12(5):4280–94.

[36] Xia W, Huang H, Duque EMS, Hou S, Palensky P, Vergara PP. Comparative assessment of generative models for transformer-and consumer-level load profiles generation. Sustain Energy Grids Netw 2024;38:101338.

[37] Hart WE, Laird CD, Watson J-P, Woodruff DL, Hackebeil GA, Nicholson BL, et al. Pyomo-optimization modeling in python, vol. 67, Springer; 2017.

[38] Salehi M, Rezaei MM. An improved probabilistic load flow in distribution networks based on clustering and Point estimate methods. Energy AI 2023;14:100272.

[39] Lin X, Zhong W, Lin X, Zhou Y, Jiang L, Du-Ikonen L, et al. Component modeling and updating method of integrated energy systems based on knowledge distillation. Energy AI 2024;16:100350.