ICLR
International Conference On Learning Representations

Offline Reinforcement Learning for Microgrid Voltage Regulation

Shan Yang, Yongli Zhu

Sun Yat-sen University

Guangzhou, China

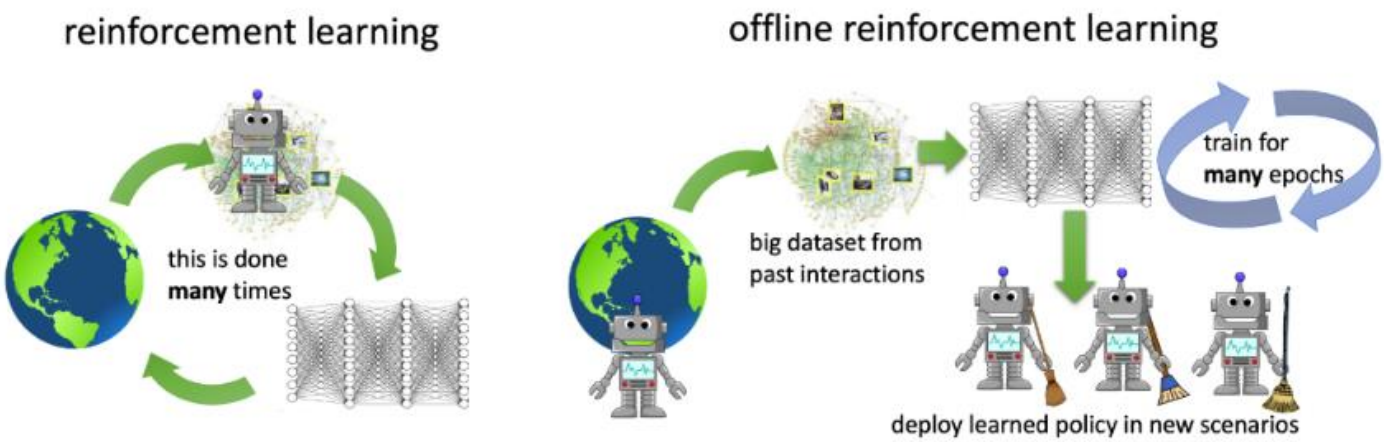
Tackling Climate Change With Machine Learning Workshop @ ICLR 2025

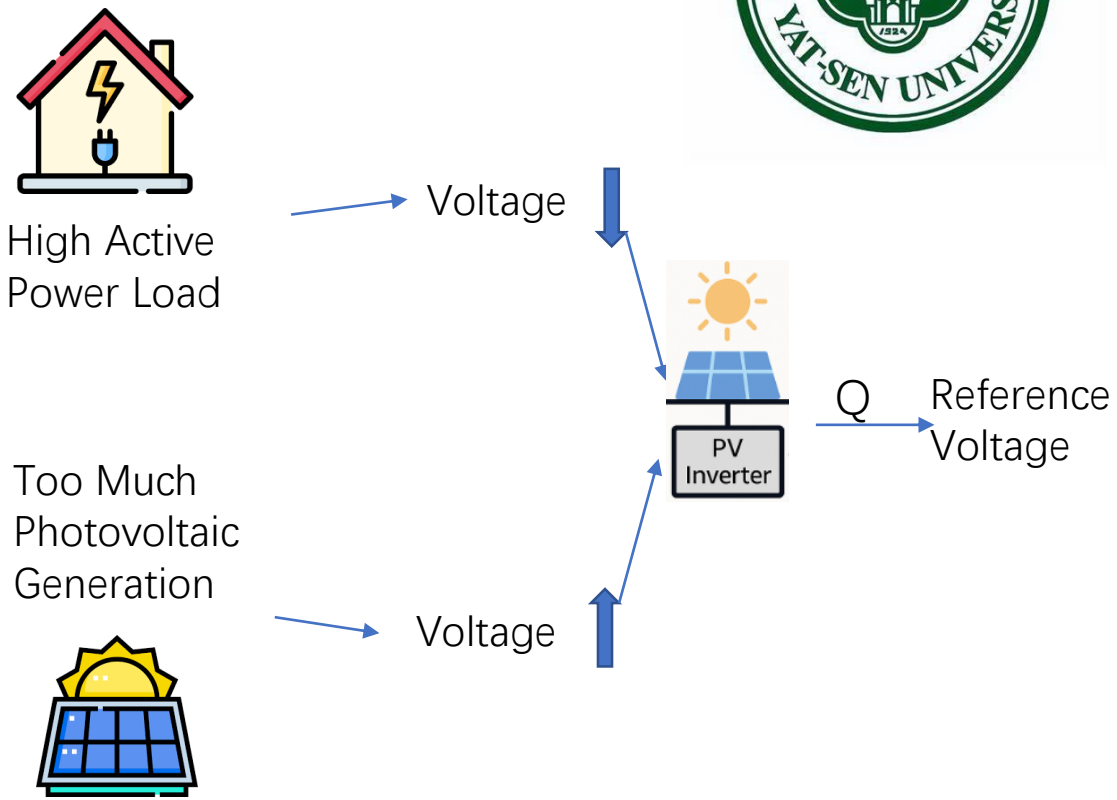
Introduction

- **Challenge:**
  Voltage regulation is challenging in renewable–penetrated microgrids due to stochastic power fluctuations.

- **Limitations of Conventional Methods:**
  Real-time control is costly and risky, especially for a microgrid with multiple renewable generators.

- **Proposed Solution:**
  Use Offline RL to train agents on existing data without real-time interaction, achieving safe and stable voltage control.



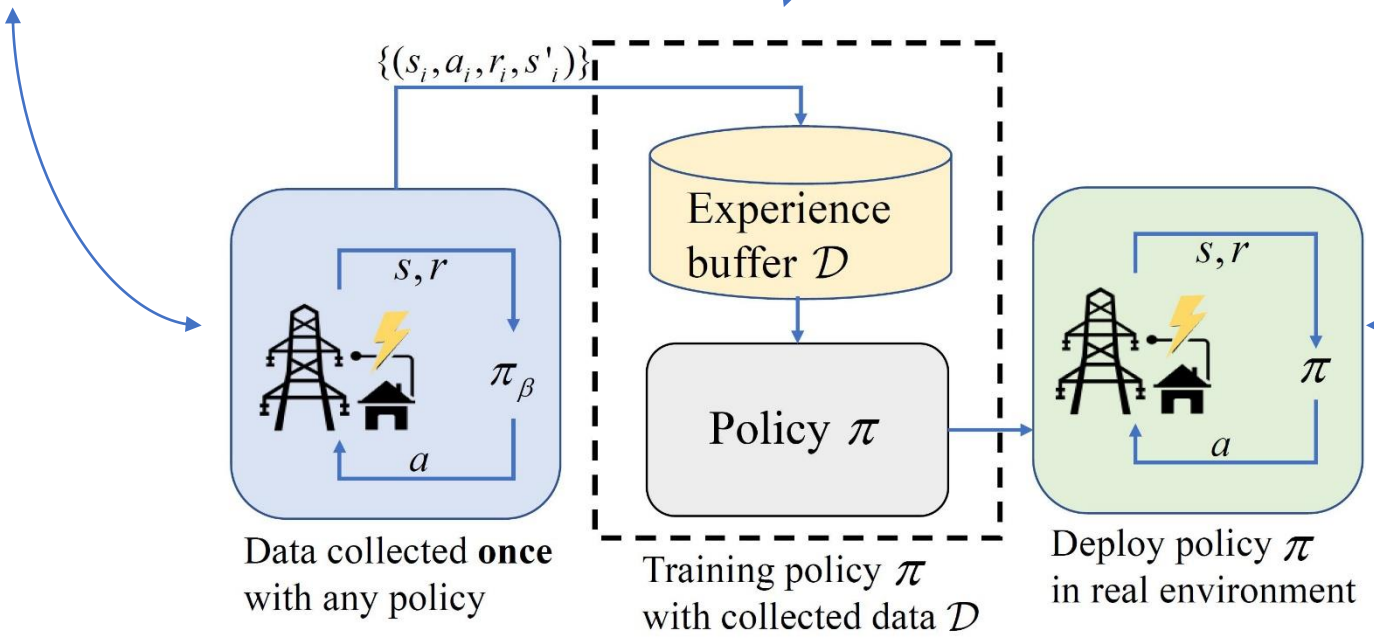


Offline Reinforcement Learning

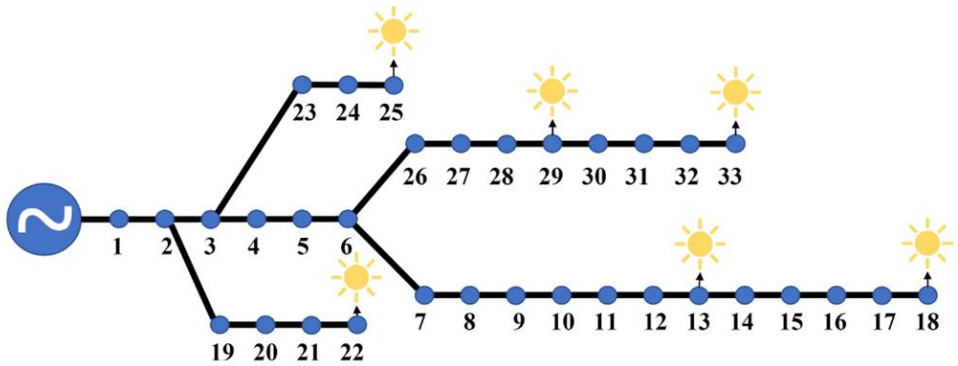
Data Collection: Data is collected **ONCE**, using any policy π_β , from the real environment. This data is stored in the experience buffer D.

Training: The policy π can **ONLY** be trained using the collected **offline** data. **No** real-time interaction with the environment is involved during the training process.

Deployment: The trained policy π is deployed in the real environment to perform critical tasks like voltage regulation.



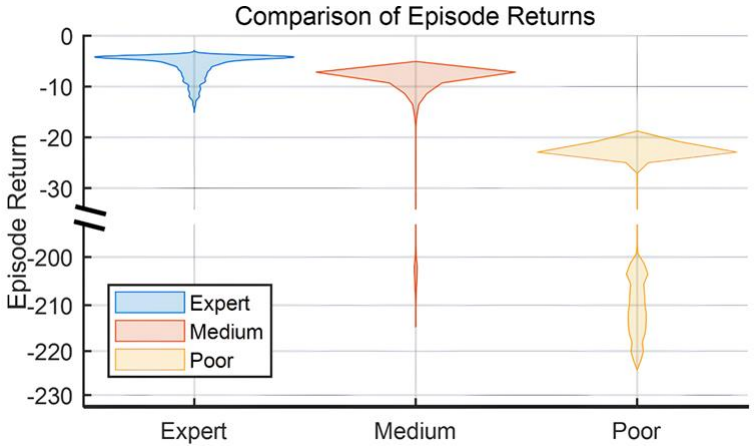
RL Environment and Datasets



The IEEE 33-bus system with solar PV penetration

RL Environment

- State (S): $S = \{p^L, q^L, p^{PV}, q^{PV}, v\}$
- Action (A): $q_k^{PV} = a_k \cdot \sqrt{(S_k^{max})^2 - (p_k^{PV})^2}$ where $-c \le a_k \le c$.
- Reward Function (r): $r = -\frac{1}{|V|}\sum_{i \in V}\left[\alpha_1 |v_i| + \alpha_2 |q_i^{PV}|\right]$
- Objective Function: $\max_\pi \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$



Dataset

- **Expert:** High-quality data from a well-trained agent.
- **Medium:** Adds 5% noise to expert actions.
- **Poor:** Random actions with low-quality transitions.
- **Each dataset:** 200,000 transitions (s,a,r,s')

Performance Metric	Expert	Medium	Poor
Average Reward	-0.0242	-0.0678	-0.4325
Reward Variance	1.97E-04	6.5995	7.6837
Reward Std. Deviation	0.014	2.5691	2.227
Average Episode Return	-5.7464	-15.8355	-85.3963
Maximum Episode Return	-2.5771	-13.75	-20.349
Return Variance	4.9976	1.44E+3	7.91E+3

Offline Reinforcement Learning

➢ BCQ Algorithm: Key Innovations

◆ **Action Constraints:**

- BCQ limits actions to those observed in offline data.

◆ **VAE and Perturbation Model:**

- Uses a VAE and perturbation network to refine actions.

$$a^* = \arg\max_a Q(s,a) + \xi(s,a)$$

Algorithm 1 Batch-Constrained Q-learning (BCQ)

Input: Batch B, horizon T, target update rate τ, mini-batch size N, max perturbation Φ, sampled actions n, weighting λ

Output: Trained networks for BCQ

Initialize $Q_{\theta_1}, Q_{\theta_2}, \xi_\phi, G_\omega = \{E_{\omega_1}, D_{\omega_2}\}$ with random parameters $\theta_1, \theta_2, \phi, \omega$
Initialize target networks $Q_{\theta'_1}, Q_{\theta'_2}, \xi'_\phi$: $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2, \phi' \leftarrow \phi$
for $t = 1$ to T **do**
  Sample N transitions (s, a, r, s') from B and compute $\mu, \sigma = E_{\omega_1}(s, a), a' = D_{\omega_2}(s, z), z \sim \mathcal{N}(\mu, \sigma)$. Update $\omega \leftarrow \arg\min_\omega \sum (a - a')^2 + D_{\mathrm{KL}}(\mathcal{N}(\mu, \sigma) \| \mathcal{N}(0, 1))$.
  Sample n actions $\{a_i \sim G_\omega(s')\}_{i=1}^n$, perturb a_i: $a_i \leftarrow a_i + \xi_\phi(s', a_i, \Phi)$.
  Compute target $y = r + \gamma \max_{a_i} \left[(1-\lambda) \max_{j=1,2} Q_{\theta'_j}(s', a_i) + \lambda \min_{j=1,2} Q_{\theta'_j}(s', a_i) \right]$.
  Update $\theta_i \leftarrow \arg\min_{\theta_i} \sum \left(y - Q_{\theta_i}(s, a) \right)^2$.
  Update $\phi \leftarrow \arg\max_\phi \sum Q_{\theta_i}\left(s, a + \xi_\phi(s, a, \Phi)\right), a \sim G_\omega(s)$.
  Update target networks: $\theta'_i \leftarrow \tau \theta_i + (1-\tau)\theta'_i$, $\phi' \leftarrow \tau \phi + (1-\tau)\phi'$.
end for

Offline Reinforcement Learning

➢ CQL Algorithm: Key Innovations

◆ Q-Function Update with Conservative Penalty:

$$L(Q) = J_{\text{CQL}}(Q) + \zeta \cdot \mathbb{E}_{(s,a,r,s')\sim D}\left[\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)^2\right]$$

◆ Action-Specific Penalty:

CQL also introduces an action-specific penalty to avoid favoring unobserved actions:

$$\pi(a \mid s) \approx \exp\left(Q(s,a) - \max_{a'} Q(s,a')\right)$$

Algorithm 2 Conservative Q-Learning (CQL)

Input: Batch B, horizon T, target update rate τ, mini-batch size N, max perturbation Φ, sampled actions n, weighting λ and ζ

Output: Trained networks for CQL

1: Initialize $Q_{\theta_1}, Q_{\theta_2}, \xi_\phi, G_\omega = \{E_{\omega_1}, D_{\omega_2}\}$ with random parameters $\theta_1, \theta_2, \phi, \omega$

2: Initialize target networks $Q_{\theta_1'}, Q_{\theta_2'}, \xi_\phi'$: $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi$

3: **for** $t = 1$ to T **do**

4: Sample N transitions (s,a,r,s') from B and compute $\mu, \sigma = E_{\omega_1}(s,a), a' = D_{\omega_2}(s,z), z \sim \mathcal{N}(\mu,\sigma)$. Update $\omega \leftarrow \arg\min_\omega \sum (a-a')^2 + D_{\text{KL}}(\mathcal{N}(\mu,\sigma)\|\mathcal{N}(0,1))$.

5: Sample n actions $\{a_i \sim G_\omega(s')\}_{i=1}^n$, perturb a_i: $a_i \leftarrow a_i + \xi_\phi(s', a_i, \Phi)$.

6: Compute target $y = r + \gamma \max_{a_i}\left[(1-\lambda)\max_{j=1,2} Q_{\theta_j'}(s',a_i) + \lambda \min_{j=1,2} Q_{\theta_j'}(s',a_i)\right]$.

7: Compute CQL penalty:

$$J_{\text{CQL}}(Q_{\theta_i}) = \mathbb{E}_{s\sim B}\left[\log \sum_a \exp Q_{\theta_i}(s,a) - \mathbb{E}_{a\sim B}[Q_{\theta_i}(s,a)]\right]$$

8: Update Q-function with combined CQL penalty and Bellman loss:

$$L(Q_{\theta_i}) = J_{\text{CQL}}(Q_{\theta_i}) + \zeta \cdot \mathbb{E}_{(s,a,r,s')\sim B}\left[\left(y - Q_{\theta_i}(s,a)\right)^2\right]$$

9: Update $\theta_i \leftarrow \arg\min_{\theta_i} L(Q_{\theta_i})$.

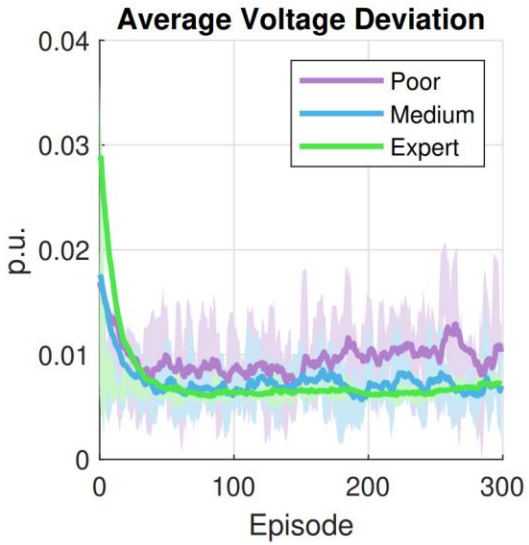
10: Update $\phi \leftarrow \arg\max_\phi \sum Q_{\theta_i}\left(s, a + \xi_\phi(s,a,\Phi)\right), a \sim G_\omega(s)$.

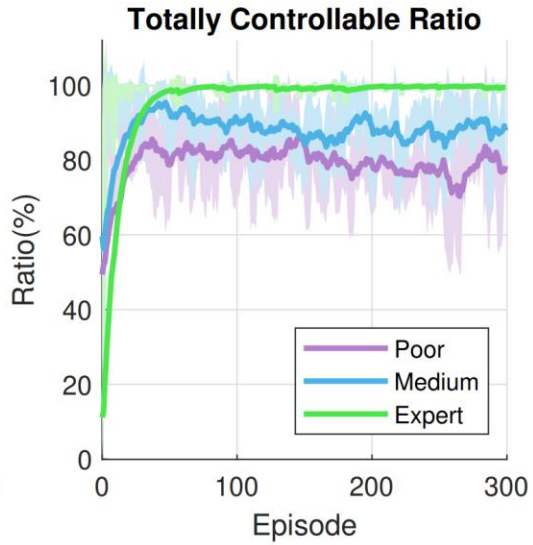
11: Update target networks: $\theta_i' \leftarrow \tau\,\theta_i + (1-\tau)\,\theta_i'$, $\phi' \leftarrow \tau\,\phi + (1-\tau)\,\phi'$.

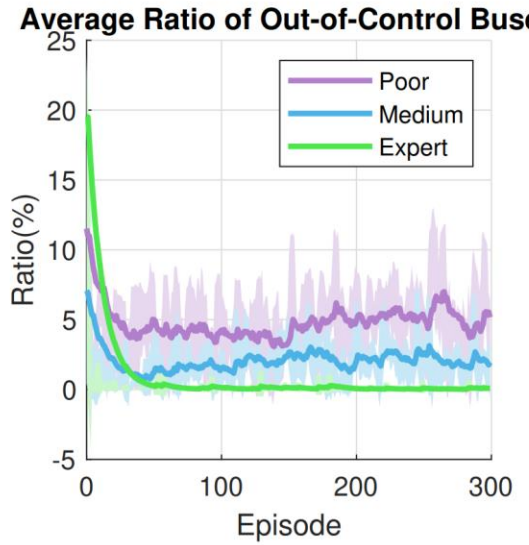
12: **end for**

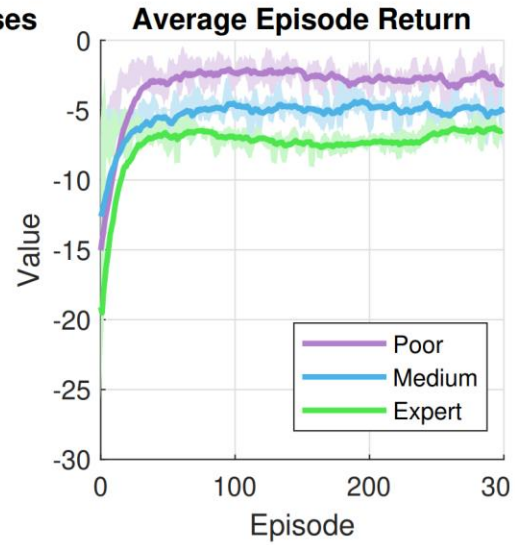
Experiment Results

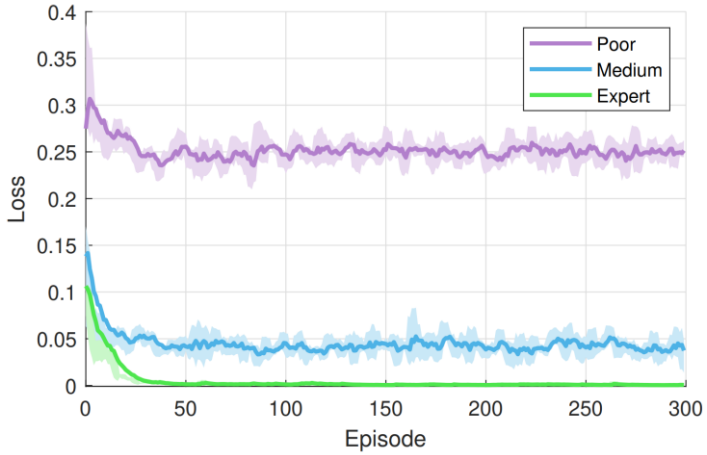
➢ Detailed Evaluation of BCQ's Performance







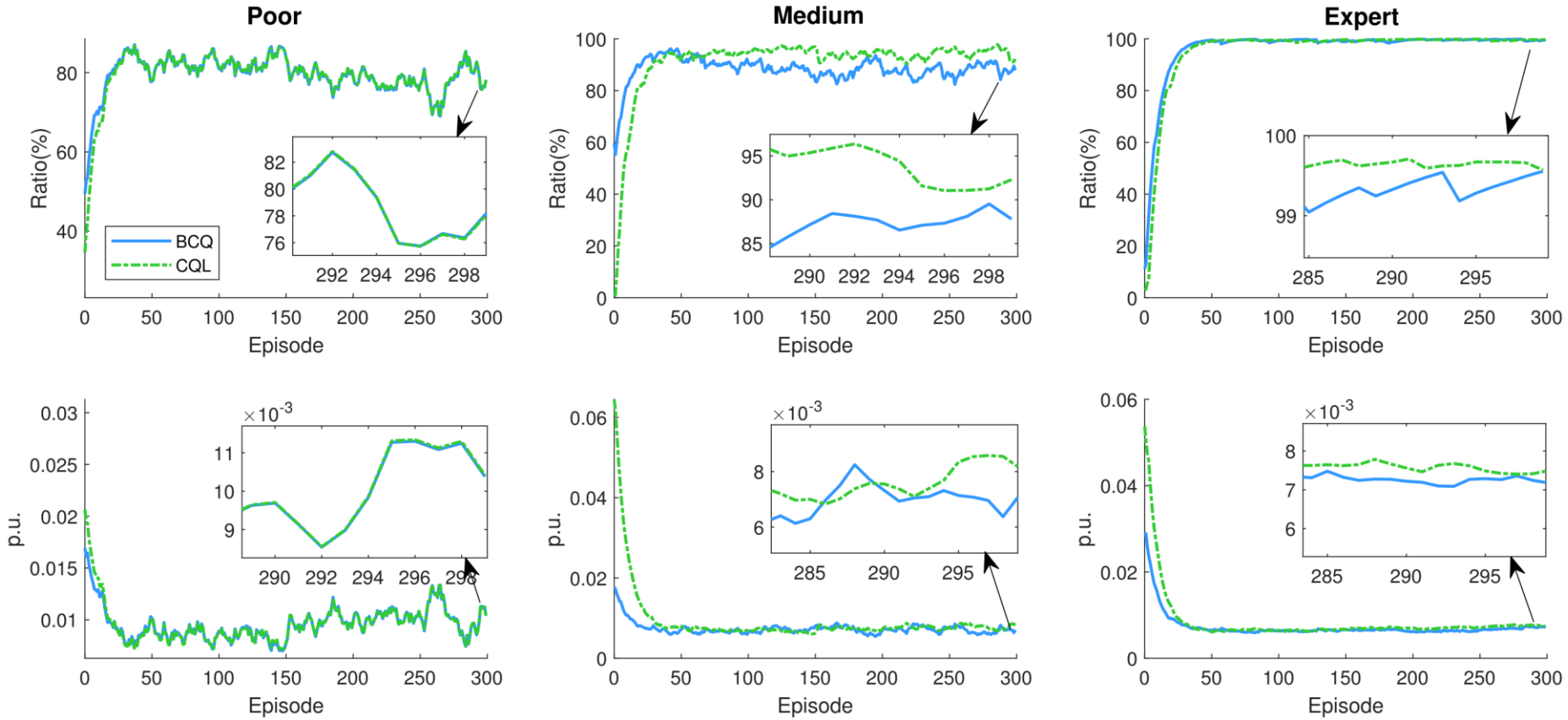




- *Voltage Deviation*: Lowest in Expert, highest in Poor.
- *Controllable Ratio*: Highest in Expert, drops with data quality.
- *Out-of-Control Buses*: More frequent in Poor dataset.
- *Episode Return*: **BCQ can still learn an acceptable strategy on Poor dataset .**

Experiment Results

➢ Performance Comparison Between BCQ and CQL



- **CQL** outperforms **BCQ** on **Poor** and **Medium** datasets, with higher controllable rate & lower voltage deviations.
- Both algorithms perform well on the **Expert** dataset, but **CQL** is **slightly more stable**.
- **CQL** is more robust, especially with low-quality data.

Conclusion & Future Work

Conclusion:

- Offline RL, particularly CQL, provides a good solution for voltage regulation in microgrids where online interaction is disallowed.

- CQL consistently outperforms BCQ in terms of stability and controllability, particularly in noisy or incomplete datasets.

Future Work:

- Apply Offline RL to more complicated power grid cases.

- Explore other Offline RL techniques to further improve the algorithmic scalability and learning efficiencies.

Thanks!

yangsh237@sysu.edu.cn,
yzhu16@alum.utk.edu

Tackling Climate Change With Machine Learning
Workshop @ ICLR 2025