SAND2022-4888C

Using Reinforcement Learning to Increase Grid Security Under Contingency Conditions

Stephen Verzi
Systems Research, Analysis and Applications
Sandia National Laboratories
Albuquerque, NM USA
sjverzi@sandia.gov
https://orcid.org/0000-0003-2152-851X

Ross Guttromson
Electric Power Systems Research
Sandia National Laboratories
Albuquerque, NM USA
rguttro@sandia.gov

Asael Sorensen
Operations Research and Computational Analysis
Sandia National Laboratories
Albuquerque, NM USA
ahsoren@sandia.gov

***Abstract*—Grid operating security studies are typically employed to establish operating boundaries, ensuring secure and stable operation for a range of operations under NERC guidelines. However, if these boundaries are violated, the existing system security margins will be largely unknown. As an alternative to the use of complex optimizations over dynamic conditions, this work employs the use of reinforcement-based machine learning to identify a sequence of secure state transitions which place the grid in a higher degree of operating security with greater static and dynamic stability margins. The approach requires the training of a machine learning agent to accomplish this task using modeled data and employs it as a decision support tool under severe, near-blackout conditions.**

Keywords— Grid Security, Reinforcement Learning, Stability, Agent Based Learning, Machine Learning

I. INTRODUCTION

During near blackout conditions, grid operators may be able to restore the system to a safe condition using a Machine Learning (ML) real-time decision support tool. Doing so would require the ability to transition (or dispatch) the grid into a 'good' state, defined as a unique dispatch where the grid has a high margin of operational security and is serving a near maximum amount of load. Under severe emergency conditions, it may not be possible to determine this operating point using optimization, nor might there be sufficient time to find it by study. Additionally, when operating outside of planned operating criteria, an improved dispatch solution requires following a path from the current state to the final state to ensure avoidance of instabilities or other detrimental operating conditions. Therefore, this paper presents an alternative approach to improve grid security through a series of incremental dispatches using Reinforcement Learning (RL). Each agent used in this research is referred to as a navigable player since it operates inside a power grid simulation environment, but more strictly it represents the decision process to change the overall grid dispatch to an improved state of security while serving as much load as possible, near-maximum if achievable. The RL method offers the potential of a speedy solution, with a high probability of achieving a secure, near-optimal solution. The RL agent selects incremental changes in the dispatch which represent a secure transition path from the current state to the final state, all the while improving upon load served when possible.

This work formalizes these transitions as a Markov Decision Process (MDP). A MDP consisting of 1) states, 2) actions to change state, 3) a transition matrix containing probabilities for transitioning from on state to another with respect to a given action and 4) immediate reward received after transitioning from one state to another. The MDP provides a formal specification for reinforcement learning in a simulation, or game, environment. In this research each agent choses an action from the current state to enact a change in the state and environment, and more specifically the RL agent learns to improve its choice of action over time. To assess the security of the grid during changes in dispatch we employed specific security metrics to ensure safe transitions during off-nominal operation, and this is a key difference between this and prior work [1-3]. The RL agent learns to navigate the MDP state space by increasing security measure values with the goal of increasing margins from catastrophic boundaries in the operational environment.

II. POWER GRID MODEL

To train and deploy the RL agent, a simplified dynamic power grid model was developed. The power system is a an asymmetric, three-area, three generator system which exhibits sufficient complexity to demonstrate the need for the RL agent to navigate in achieving its objective of increasing operational security and improving load served, while avoiding instabilities. Three control dimensions define the state space; these are the dispatch outputs of generators 1, 2 and 3. Total system load was automatically scaled to ensure it was equal to total system generation plus losses for each dispatch state. Operational security metrics were determined using Power System Toolbox (PST) [4] analysis running on MATLAB® and imposed onto each discrete state. Security metrics include minimum and maximum bus voltages, minimum critical clearing time, maximum line flow, and minimum damping ratio. The objective of each agent is to maximize the system load served while maintaining a sufficient margin of operational security.

A. Grid Model and Topology

The grid topology utilized for this work is shown in Fig. 1. A three-dimensional discrete state space was developed of size 25^3. The RL agent's objective is to incrementally adjust the system dispatch in a manner that increases system load while simultaneously improving system security.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND No. 2022-XXXX C.

978-1-6654-6591-5/22/$31.00 ©2022 IEEE

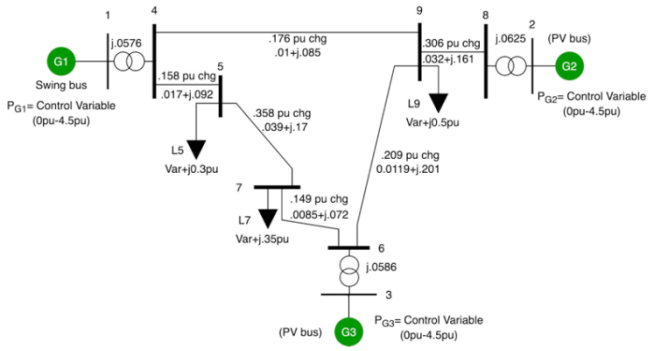

Fig. 1. System Toplogy for Agent Navigation and Training.

System security metrics were made available to navigating agents for limited feedback during the navigation process. Fig. 2 shows a 2-D slice of this state space, holding Generator 3 power output constant at 1.0 pu. Time domain simulation was replaced with eigen-analysis and the use of equal area criterion [5, 6]. The upper region of the panels in Fig. 2 (marked with x's) did not result in converged load flows, and were interpreted as unsecure dispatches, whereas dots represent converged load flows.

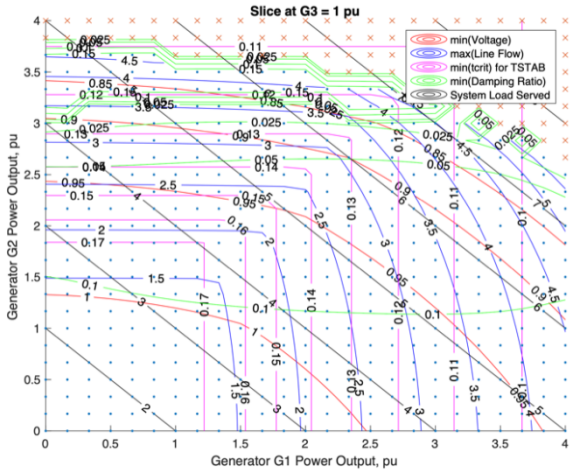

Fig. 2. Two-Dimensional Slice of a 3-D State Space (Disaptch) for the Power System Shown in Fig. 1.

The following bulleted list describes the computation of each security measure.

- **Bus Voltage**: The voltage contours shown in Fig. 2 represents the minimum of all nine bus voltages as determined by load flow for each state. There were no conditions that resulted in overvoltage conditions for this system.
- **Line Flow**: The line flow contours are represented as the maximum line flow for all lines at each state, as determined by a load flow analysis.
- **Transient Stability**: Critical clearing time was calculated for each generator, for each state, using a Thevenin system reduction.
- **Small Signal Stability**: For each state, the dynamic grid models were linearized, eigenvalues found, and damping ratios calculated.

Fig. 2 is decomposed into four panels, shown in Fig. 3. Within each panel of Fig. 3, the green region represents adequate system security, the yellow region represents stable but poor grid security, and the red region represents unacceptable system security.

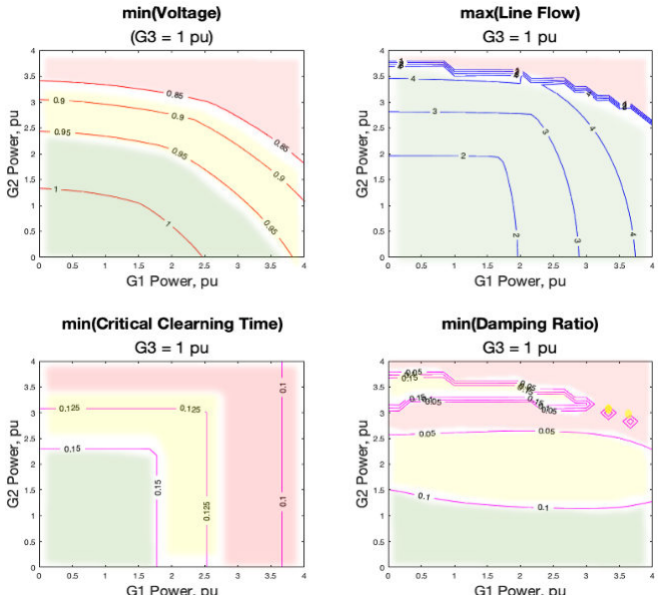

Fig. 3. Constraints Shown Indiviually Here Are Deconstructed from Fig. 2. Colors Identify the Reward Regions Defined in Table I. Green Represents a Feasible Region, in which the Optimal Exists. Yellow Regions Are Penalty Regions where Operation Is Possible but Undesired. Red Represents Instability or Failure.

The boundaries for these regions are defined mathematically in Table I.

TABLE I. SECURITY REGIONS: FEASIBILITY, PENALTY AND FAILURE

Security Metric	Security Condition	Type of Region
Bus Voltage	0.95 pu ≤ min(Vbus) *-and-* max(Vbus) ≤ 1.05 pu	feasibility
Line Flow	Line_Flow ≤ 4 pu	feasibility
Transient Stability	125 ms ≤ min(tcrit)	feasibility
Small Signal Stability	7 % ≤ damping_ratio	feasibility
Bus Voltage	0.85 pu ≤ min(Vbus) < 0.95 pu *-or-* 1.05 pu < max(Vbus) ≤ 1.15 pu	penalty
Line Flow	4.0 pu < max(line flow) ≤ 4.5 pu	penalty
Transient Stability	100 ms ≤ min(tcrit) < 125 ms	penalty
Small Signal Stability	1.5 % ≤ damping_ratio < 7 %	penalty
Bus Voltage	min(Vbus) < 0.85 pu *-or-* 1.15 pu < max(Vbus)	failure
Line Flow	4.5 pu < max(line flow)	failure
Transient Stability	min(tcrit) < 100 ms	failure
Small Signal Stability	damping_ratio < 1.5 %	failure

B. Reward Region and Security Condition Specification

The security constraints shown in Fig. 2 were converted to security condition equations for use by agents during navigation and learning. These equations were derived as listed in Table I, such that the instantaneous MDP reward for navigation in feasibility regions is larger than that for penalty regions which in turn is larger than that of failure regions. The term feasibility, shown in Table I was selected as an analogy to an optimization problem. In all cases, the optimal state, the state with the highest reward, is state within the feasible region that is serving the highest amount of system load.

C. Grid, State Space Dimensionality

The viability of the technique presented is dependent upon the ability of the RL agent to successfully navigate higher dimensional spaces. These dimensions arise from the independent control of additional generators, loads, and topological configurations. Although this paper is considering a simplified system with only three control dimensions, higher dimensional systems are currently being developed which utilize transfer learning and the use of variable navigation control step size(s).

III. MACHINE LEARNING MODEL

In this research, we employ deep reinforcement learning (DRL) to learn appropriate actions to take for a given grid state determined using the data samples generated by the PST model described previously.

A. DRL Description

For each episode, the RL agent is initially randomly placed in a penalty region of the state space as defined in Table I and must learn to navigate using a pre-defined number of state transitions to improve grid security and load served. Each step of the sequence is constrained to one discrete transition (increase or decrease) in a single generator or to maintain status quo. During training the RL agent will use reward signals received for actions taken from the current state to update its navigation policy, indexed by state, action pairs. Through this training process, the RL agent learns to navigate according to the resulting MDP transition matrix, which is enforced as part of the MDP environment and yet not directly visible to any navigating agent.

B. RL Agent Model Structure

For learning the RL agent uses q-learning [7] implemented using a deep neural network, also called a Deep Q-Network (DQN) [8]. The RL agent model and software for this research was extended from Bailey and colleagues for navigation in a transmission grid environment [9]. The final score for each navigation trajectory (also called episode) of length n is determined using Eq. (1).

$$\sum_{i=1}^{n} Q(S_i,A_i) \tag{1}$$

where $Q(S_i,A_i)$ is the approximated expected reward for taking action A_i while in state S_i. The approximate expected reward is updated during learning according to Eq. (2).

$$Q(S_i,A_i) = Q(S_i,A_i) + \alpha\left[R(S_{i+1}) + \gamma \max_a Q(S_{i+1},a) - Q(S_i,A_i)\right] \tag{2}$$

where α is the learning rate, γ is the discount rate, $R(S_{i+1})$ is the reward received at step $i+1$ (arriving at state S_{i+1}) after taking action A_i from state S_i, and a ranges over the actions that can be taken at step $i+1$ when the environment is at state S_{i+1}. Thus, the RL agent learns to account for expected future reward discounted by γ. The immediate reward ($R(S_i)$) for arriving at state S_i resulting from taking action A_{i-1} while in state $S_{i-1}$is shown in Eq. (3).

$$R(S_i) = \begin{cases} \frac{f(S_i)l(S_i)}{\max_j f(S_j)l(S_j)} + f(S_i) & \text{if } f(S_i) = 1 \\ \frac{f(S_i)l(S_i)}{\max_j f(S_j)l(S_j)} & \text{otherwise} \end{cases} \tag{3}$$

where $f(S_i)$ is the security condition (either feasible, penalty or failure) and $l(S_i)$ is the load served for S_i. When S_i is in the feasible region (see Table I and Fig. 3), $f(S_i) = 1$, and when S_i is in the failure region, $f(S_i) = 0$. For S_i in the penalty region, $f(S_i)$ ranges linearly between 0 (where it touches the failure region) and 1 (where it reaches the feasible region). Note that if a failure state (defined in Tablel I), is reached due to an action choice, then navigation is stopped and no further reward received. Navigation is not stopped when the maximal reward is received, as part of this research is to train the RL agent to recognize when to stop and also to minimize the number of actions taken during navigation. This reward serves as the objective for the RL agent navigation during learning and is also used by the greedy agent for its navigation. The reward space for the same slice shown in Fig. 2 and disaggregated in Fig. 3 is shown in Fig. 4.

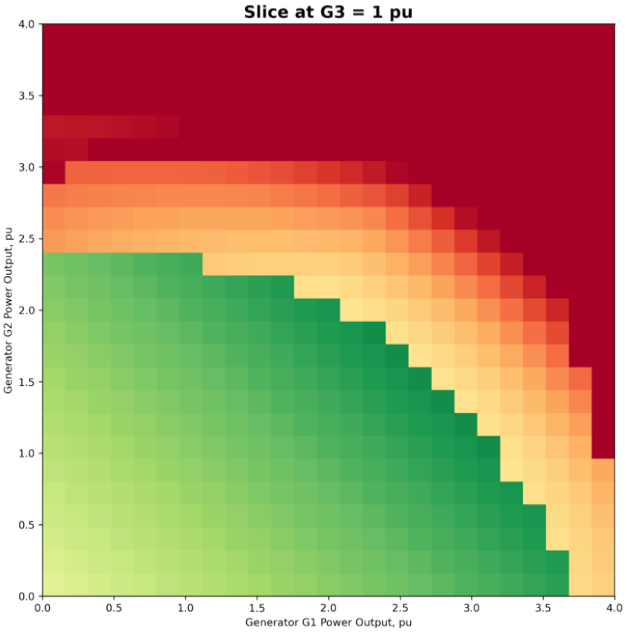

Fig. 4. Two-dimensional Slice (at G3 = 1 pu) of Reward Space for Navigation, where Unstable Is Shown in Red (with Reward Value of 0), Feasibility Is Shown in Green (with Feasibility of 1 and Reward Value between 0.5 and 1.0) and All Other States in Yellow Are Used as Starting States (with Reward Values Between 0 and 0.5, Inclusive).

The feasibility condition, $f(S_i)$ in Eq. (3) and shown as part of the reward in Fig. 4, is defined using the minimum of each of the operational security measure penalty conditions: bus voltage (BV), line flow (LF), transient stability (TS), and small signal stability (SSS), defined in Table I. Thus, the values shown in Fig. 4 are composed from the minimum of values shown in Fig. 3 multiplied by the load served, $l(S_i)$, and normalized once again to range between 0 and 1, inclusive. Note that the penalty conditions (see Table I) are normalized such that they range between between 0 and 1, inclusive, individually, where feasibility occurs when the combined penalty is equal to 1 and unstable dispatch conditions have a value of 0. Fig. 4 shows a 2-D slice of the reward space environment within which the agents navigate, matching the conditions used in Fig. 2 and Fig. 3.

A random starting location was selected for each episode during testing. The same initial state value was use used for both Greedy and Random agents as well. Tests were conducted using all possible starting states for results presented in this paper. All starting locations were chosen from within the 3-D penalty area identified in Table I (and shown in yellow in Fig. 4). During training, each starting location was randomly chosen to prevent the RL agent from over-training on any subset of the entire set of starting states. It also prevents the game from ending before any agent's first step, as it would if the initial condition were placed in the unstable region.

C. ML Model Implementation

A simple feedforward neural network architecture consisting of two dense layers was used in the RL agent DQN model. The RL agent is implemented in python using the PyTorch deep learning libraries [10] on three different GPU systems.

Following DQN learning, approximate expected reward is defined in Eq. (3). To promote exploration ε-greedy is used, defined as, $\varepsilon_i = \varepsilon_{\min} + (\varepsilon_{\max} - \varepsilon_{\min})e^{-\frac{i}{\delta}}$ for each episode i during training, which means that there will always be at least $\varepsilon_{\min}$ stochastistity (or randomness) in action choice [11]. We did not optimize network structure or hyperparameters for this effort, but these will be addressed in future work. The RL agent model parameters are listed in Table II.

TABLE II. RL AGENT MODEL PARAMETERS

Parameter	Value
γ (Discount Rate)	0.8
α (Learning Rate)	0.00025
$\varepsilon_{\min}$ / $\varepsilon_{\max}$	0.1 / 0.9
δ (ε Decay Rate)	1000
Replay Memory	50,000
Target Update Delay	50

For more details on this DQN algorithm and software, see Bailey and colleagues [9].

IV. METHOD OF RL AGENT TRAINING

The RL Agent is trained using 15,000 navigation episodes consisting of a maximum of 50 steps (or actions). A rolling replay memory buffer of 50,000 episodes is maintained, from which 1,024 samples are randomly chosen for training. Policy target update is delayed by 50 episodes (i.e., the RL agent policy is re-trained after 50 episodes).

A. Benchmark Comparisons

To measure how well the trained RL agent performs, it is compared it to both Random and one-state-look-ahead Greedy agents. The Random agent serves as a lower bound for comparison to the RL agent. The Random agent is constrained to make only a single discrete state transition per step. The Greedy agent uses the reward information for the current state as well as all neighboring state rewards to form its Decision Policy, i.e., it will always navigate to the next state with highest reward, even if it means staying put. We expect that the RL agent should approach, and possibly even surpass the Greedy agent's performance given enough training and sufficient grid environment complexity.

V. RESULTS AND CONCLUSIONS

Table III contains comparison results for navigation in the transition grid environment for the trained RL agent, the Random agent and the Greedy agent across 5,180 episodes (one for each possible starting state dispatch in the penalty region defined in Table I) where there is no further learning occurring in the RL agent after training.

TABLE III. FEASIBILITY-UNSTABLE-PENALTY PERCENTAGE COMPARISON FOR ALL AGENTS

The results in Table III for the RL agent are after 15,000 episodes of training, which is not sufficient to match the Greedy agent, even for this simple environment. During learning, the RL agent improves is cumulative reward value. In these results, the RL agent can achieve 93.08% of the maximal feasible servable load as compared with 83.74% and 74.06% for the Greedy and Random agents, respectively. Fig. 4 shows a histogram for all three agents with respect to cumulative reward across all trajectories during testing.

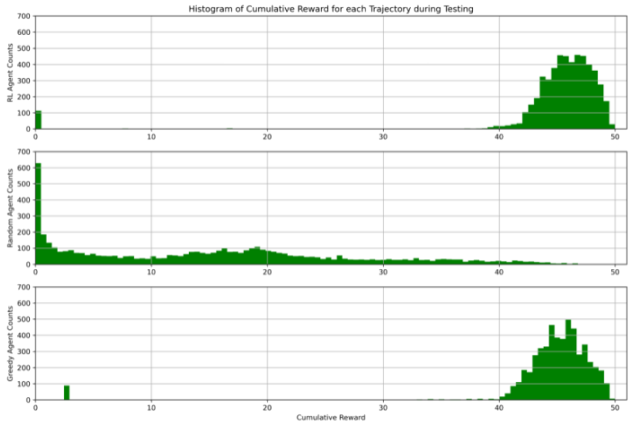

Fig. 5. Cumulative reward histograms for each agent (RL – top, Random – middle and Greedy – bottom) during testing (5,180 episodes).

An example trajectory for each of the three agents is shown in Fig. 6. In this example, both Greedy and RL agents navigate from a starting dispatch state (shown with an open circle) to a safer dispatch state (shown with an "X") moving to feasibility (color changing from red to green) along the way, while increasing security and load served. The Random Agent moved about randomly for a while before encountering an unstable state, at which point navigation is stopped. As the agent transitions, its reward at each state is proportional to the grid security at that state.

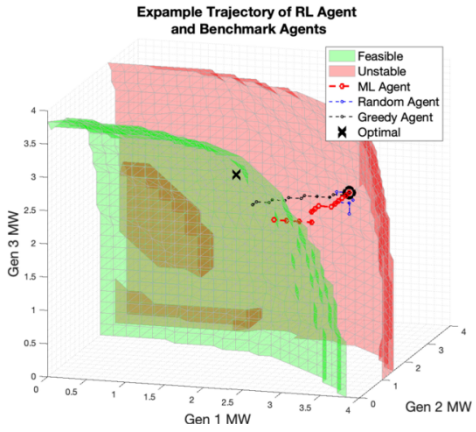

Fig. 6. Cumulative reward histograms for each agent (RL – top, Random – middle and Greedy – bottom) during testing.

Agent	Feasible %	Unstable %	Penalty %
Random Agent	18.15	44.98	36.87
RL Agent	97.49	2.39	0.12
Greedy Agent	98.30	0.00	1.70

VI. FURTHER WORK

Future work includes expanding the dimensionality, use of model-based approaches, and moving to continuous action spaces for generation control.

ACKNOWLEDGMENT

The authors would like to acknowledge project support and sponsorship by Dr. Ali Ghassemian of the Department of Energy's Office of Electricity Advanced Grid Modeling program.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. SAND No. 2022-XXXX C.

REFERENCES

[1] Y. Zhou, B. Zhang, C. Xu, T. Lan, R. Diao, D. Shi, Z. Wang, and W. Lee "A data-driven method for fast AC optimal power flow solutions via deep reinforcement learning," Journal of Modern Power Systems and Clean Energy, vol. 8, no. 6, pp. 1128-1139, 2020. https://doi.org/10.35833/MPCE.2020.000522

[2] Y. Zhang, J. Wu, Z. Chen, Y. Huang, and Z. Zheng, "Sequential node/link recovery strategy of power grids based on Q-learning approach," in 2019 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5, 2019. doi: 10.1109/ISCAS.2019.8702107

[3] M. Mandich, "Power System Stability Assessment with Supervised Machine Learning," Masters thesis, University of Tennessee, 2021.

[4] Power System Toolbox. (2008). Cherry Tree Scientific Software.

[5] J. D. Glover, M. S. Sarma, and T. Overbye, Power System Analysis & Design, SI Version. Stamford, CT: Cengage Learning, 2012. ISBN: 978-1-111-42579-1

[6] P. Kundur, Power System Stability and Control. New York, NY: McGraw-hill, 1994. ISBN: 978-0-070-35948-1

[7] C. J. Watkins and P. Dayan, "Q-learning," Machine learning, vol. 8, no. 3-4, pp. 279-292, 1992. https://doi.org/10.1007/BF00992698

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

[9] T. Bailey, J. Johnson, and D. Levin, "Deep reinforcement learning for online distribution power system cybersecurity protection," in 2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pp. 227-232, 2021. https://doi.org/10.1109/SmartGridComm51999.2021.9631991

[10] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," Advances in neural information processing systems, vol. 32, pp. 8026-8037, 2019.

[11] R. S. Sutton, and A. G. Barto, Reinforcement Learning: An Introduction, Cambridge, MA: A Bradford Book, 2018. ISBN: 978-0-262-03924-6