Efficient Reinforcement Learning for Real-Time Hardware-Based Energy System Experiments

Alexander Stevenson[1,2], **Mayank Panwar**[1], **Rob Hovsapian**[1], **Arif Sarwat**[2]

[1]National Renewable Energy Laboratory, Golden CO, USA
[2]Department of Electrical and Computer Engineering, Florida International University, Miami FL, USA
alexander.stevenson@nrel.gov, mayank.panwar@nrel.gov

Abstract

In the context of urgent climate challenges and the pressing need for rapid technology development, Reinforcement Learning (RL) stands as a compelling data-driven method for controlling real-world physical systems. However, RL implementation often entails time-consuming and computationally intensive data collection and training processes, rendering them inefficient for real-time applications that lack non-real-time models. To address these limitations, real-time emulation techniques have emerged as valuable tools for the lab-scale rapid prototyping of intricate energy systems. While emulated systems offer a bridge between simulation and reality, they too face constraints, hindering comprehensive characterization, testing, and development. In this research, we construct a surrogate model using limited data from simulated systems, enabling an efficient and effective training process for a Double Deep Q-Network (DDQN) agent for future deployment. Our approach is illustrated through a hydropower application, demonstrating the practical impact of our approach on climate-related technology development.

Introduction

There is an increasing urgency in the energy sector towards continually developing new and existing technologies that can help to reduce overall carbon emissions. Towards this, Rapid Control Prototyping (RCP) allows for cost effective, fast-paced development as opposed to full-scale experimentation. Digital Real-Time Simulation (DRTS) platforms allow the rapid prototyping of control systems by introducing real-time, low latency interaction between simulated plant model and real-time controller under development(Panwar et al. 2013).

In the last few years, RL has emerged as a practical data-driven approach for optimally controlling power system applications. However, the RL approach poses several challenges during the training and deployment such as cost and efficiency of data collection, training, computation time and experimenter's effort. Thus, efficient training of RL agents can prove useful. When first introducing untrained RL agents into a time-dependent and continuous system, it may be difficult to properly train and act as the environment may have many states where instabilities may arise, and training may need to be restarted from a stable point of time. This can be a tedious process and will likely not be very feasible. Towards this, non-real-time surrogate models may allow for an easier training experience before using a pre-trained agent in a time continuous environment.

Background and Related Work

The hydropower emulation platform presented in this work is a combined hardware and real-time software-based setup that faithfully represents hydro turbine shaft dynamics (speed, torque) using a physics-based real-time plant model and physical Variable Frequency Drive (VFD) driven dual Induction Machines (IM) (Poudel, Panwar, and Hovsapian 2023). The plant dynamics can be real-time signals generated from a physics-based or data-driven simulation model, or real-time data streams from an actual hydropower plant (Panwar 2022). The objective of hydropower emulation is to enable a hardware-based experimental platform that can be used for RCP and can drastically reduce development time and costs for hydropower technologies(Panwar et al. 2013), (Panwar 2022). However, when using physics-based models to drive physical hardware, there may be some complex dynamics that need to be addressed which may decrease emulation fidelity. Optimal Control (OC) is one possibility to reduce emulation error, however, relies on precise mathematical models within an optimization framework to determine control law. Recent developments in Model Predictive Control (MPC) for fast real-time control in real-world applications have shown promise for safety critical applications (Hewing et al. 2020). In contrast, RL uses an agent to maximize rewards in an environment through trial and error. While OC optimizes predefined objectives, which might be suboptimal due to model limitations and changing conditions, RL agents have the flexibility to discover and optimize better objectives, making them more suitable for complex and dynamic environments (Song et al. 2023).

Thus, power plant emulation can be assisted using the help of several state-of-the-art RL algorithms seen in (Lazaridis, Fachantidis, and Vlahavas 2020), which are summarized with key differences, applications, and limitations. Specifically, Deep Q-Network (DQN) learning is a model-free RL approach where success has been seen where a discrete action space is appropriate (Stevenson, Tariq, and Sarwat 2023). DQNs can be further improved in several ways

as seen in (Hessel et al. 2018), including using noisy DQNs, distributional DQNs, DDQN, and dueling DDQNs. To avoid overestimation bias and improve training, a DDQN agent architecture can be utilized to reduce error for hydropower emulation by modifying setpoint signals from a real-time model before sending them to physical hardware. The use of Hardware-in-the-Loop (HIL) techniques in developing control algorithms for real-world systems (Panwar et al. 2013),(Khalid, Stevenson, and Sarwat 2021),(Kollmer et al. 2018) is a popular way to train RL-based models for deployment due to the ability to avoid high development costs. However, depending on the role of the RL agent, training in real-time using live data can be cost prohibitive and lead to poor training. Thus, reduction in complexity of training can be accomplished using non-real-time data-driven machine learning surrogate models of complex systems, such as in (Angione C 2022), before moving to live time continuous systems. This approach of utilizing a surrogate model for pretraining the RL agent helps to bridge the "reality gap" problem of RL algorithms as described in (Li et al. 2023) for power systems implementations. As such, (Abid 2022) identifies multiple artificial intelligence-based surrogate modeling techniques used for energy system digital twin modeling including least squares, Inverse Distance Weighting (IDW), kriging, Radial Basis Functions (RBF), Artificial Neural Networks (ANN), and Support Vector Regression (SVR), with the most implemented being ANNs and kriging. ANNs have the drawback of needing more extensive data for surrogate modeling but show high performance in systems with nonlinear dynamics, thus this approach will be used. Furthermore in (Chaturvedi et al. 2023), an ANN is shown as being a successful surrogate model for training RL agents in an effective and efficient manner. Therefore, by using a data-driven surrogate hydropower plant model to efficiently train a DDQN agent for error reduction in real-time emulation, future hydropower emulation fidelity using physical equipment can be increased during RCP. This surrogate model can be considered a digital twin if it is continually updated and tuned using data from actual physical hydro plant in the field. Specifically in this work, simulation data is used to train (offline) a surrogate model which can be considered a digital model, digital shadow, or a digital twin (Adam 2022). Other methods such as Physics Informed Neural Networks (PINNs) can also be used (Raissi, Perdikaris, and Karniadakis 2019), however in this case, the focus is not on improving the dynamical representation of hydropower as a digital surrogate model, but instead to accurately extend it to actual physical emulation hardware. Any digital representation (physics-based, data-driven, digital twin, etc.) will still need to drive the emulation hardware which has unmodeled system dynamics where RL can be used to improve the emulation accuracy and response.

The paper is organized as follows: Sec. II discusses problem formulation and surrogate model-based training; Sec. III describes the hydropower plant surrogate model and training; Sec. IV describes the DDQN RL agent, it's environment and training process; Sec. V presents results and a brief discussion; Sec. VI presents the conclusions and future work.

Surrogate Model-Based Approach

For successful and smooth development and deployment of RL algorithms to a real-world environment, training must take place ideally in an environment that is forgiving, thus allowing the update of an agent such that a physical environment is not disturbed or damaged. Online training in both simulated and emulated hardware environments can provide this safe environment for training. However, when dealing with fast action, high sample rate RL agents, it may be difficult to properly control an environment consisting of a physical real-time system. Taking an action and observing its effect may be asynchronous, or even delayed, making it difficult to train using real-time data. Conversely, fast dynamics of physical system may impose requirements of costly hardware for high baud-rate data and RL training. Thus, to increase the efficiency of RL training, a surrogate model-based approach is developed and used as a precursor to online training with real-time data streams. The steps to implementing RL for hydropower emulation, and the process of using a surrogate hydropower plant model to train this RL agent are shown in Figure 1.

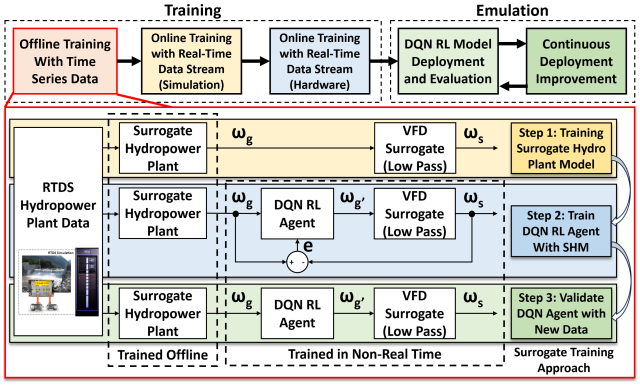

Figure 1: Surrogate model-based approach for efficient RL

This approach of pre-training an RL agent with a non-real-time environment first involves selecting and training appropriate surrogate models. In this work, an ANN acts as a hydropower plant model and a low pass filter is used to represent surrogate emulation hardware (VFD and IM). By using an ANN-based surrogate model, it is possible to learn non-linear and site-specific dynamics and quickly emulate without the need for making time-expensive simulation models.

Problem Formulation

As a continuation of prior works (Poudel, Panwar, and Hovsapian 2023), (Panwar 2022), the physical emulation of a hydropower system will be achieved using a mechanical drive system with two IMs coupled through a mechanical shaft, and a driving VFD. One IM emulates the hydro turbine by following speed setpoints provided by simulation, driven by the VFD. The other IM emulates the hydro generator, and also driven by the VFD to provide counter torque through the mechanical shaft (Poudel, Panwar, and Hovsapian 2023). The problem with sending simple simulation

signals directly to drive the IMs, are the communication delays, losses and non-linearities associated with the mechanical system that are either inaccurately represented in the simulation model or are missing entirely. By placing an RL agent between the simulation signal and the receiving VFD (IM driver), mechanical losses and non-linearities may be accounted for through signal modification. Ideally, with the correct RL mapping (from proper training) from environment variables to signal modification (a), the following Eq. (1) should be true in regard to Figure 2.

$$\omega_{g'} = \omega_g + a \ \ni \ \omega_g = \omega_s \ \forall \ t \tag{1}$$

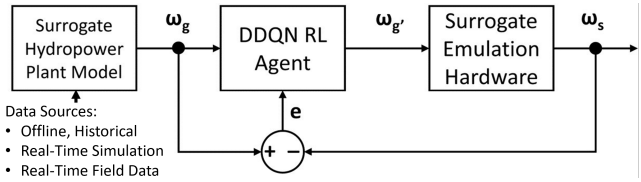

Figure 2: RL implementation into the surrogate environment

During training, the RL agent would receive the error (e) between ω_g and ω_s to determine the correct actions to take by updating the policy (discussed more in Sec. IV).

Hydropower Plant Surrogate Model

Model Architecture and Input Features

The utilized ANN model for the surrogate hydropower plant is a Long Short-Term Memory (LSTM) based neural network. The surrogate model architecture consists of a single input, single hidden LSTM, and output layer. The LSTM layer is made up of 64 units and a Rectified Linear Unit (ReLu) activation. The surrogate model takes as an input and predicts step ahead output values of a hydropower plant based on the 7 measurements taken from a physics based real-time hydropower plant model. These parameters are generator speed (ω_g), generator speed reference (ω_g^{ref}), generator power (P_g), stator power (P_{st}), stator power reference (P_{st}^{ref}), gate position (G), and turbine water flow (Q_{tur}). Thus, the surrogate model uses 7 inputs to predict 7 values for the next time step.

Training

The data used to train the surrogate model was produced by varying the power setpoint for hydropower plant model as a step change. In total, 20 step change events were recorded including 10 pumping-mode events, and 10 generating mode events. For each event, the 7 inputs to the surrogate model were sampled at a rate of 20 samples/sec (50ms per sample) for 20 seconds with a 5% pre-event trigger. Using 90% of the produced events for training, the surrogate model was trained such that the current timestep with 7 input features was used to predict the next timestep of 7 outputs. The surrogate model was then validated on the last 10% of produced data after each training epoch on pumping and generating events. The results for 2000 epochs of training is seen in Figure 3.

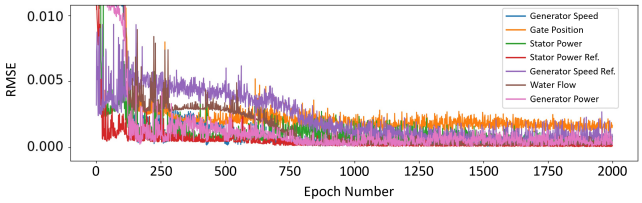

Figure 3: Results from surrogate model training

The final surrogate model produced Root Mean Squared Error (RMSE) values of 0.00023, 0.00090, 0.00035, 0.00025, 0.00024, 0.00161, and 0.00046 for ω_g, ω_g^{ref}, P_g, P_{st}, P_{st}^{ref}, G, and Q_{tur} respectively. The outputs of the trained surrogate can be seen validated for a generating event in Figure 4 which demonstrates the ability of the surrogate model to represent multiple output signals.

Implemented DDQN Agent

Observations, Actions, Policy, and Reward Function

A DDQN agent uses operational parameters, or observations, taken from its environment (surrogate models) as input features to take appropriate action. In this case, hydropower generator speed from the surrogate hydropower plant model and surrogate emulation hardware (ω_g and ω_s, respectively) are used as environment observations. After training, the DDQN will have converged to a correct set of state-actions pairs, thus mapping environment states to appropriate actions. In this application, the DDQN's action space consists of 101 discrete actions (a) to increase or decrease ω_g before sending the modified signal, $\omega_g^{'}$, to the surrogate emulation hardware as determined by (2).

$$\omega_g^{'} = \omega_g + (a \cdot i) - \frac{n \cdot (i-1)}{2} - i \tag{2}$$

Here, a is the action taken by the DDQN. i is the integer step the DDQN is allowed to increase or decrease ω_g by, in this case 0.0001 (p.u.). Lastly, n is equal to the total number of discrete actions possible, in this case 101. The second term and third term combined in Eq. 2 sets the middle of the action space to output 0, thus no modification is done to ω_g for that timestep. Anything above or below this middle action will either increase or decrease ω_g accordingly. The DDQN's policy consists of LSTM with 16 units and a dense layer with 32 processing elements and a ReLu activation function. DQN is based on the Bellman equation for Q-values as seen in (3).

$$Q_{(s,a)} = (1-\alpha) \cdot Q_{(s,a)} + \alpha \cdot (r_{(s,a)} + \gamma \cdot max_{a'} Q_{(s',a')}) \tag{3}$$

In this equation, $Q_{(s,a)}$ represents the Q-value for taking action a in state s. α is the learning rate. $r_{(s,a)}$ is the immediate reward obtained after taking action a in state s. γ is the discount factor. s' represents the next state. $max_{a'}\ Q_{(s',a')}$ is the maximum Q-value among all possible actions a' in s'. This forms the foundation for updating the Q-values in the

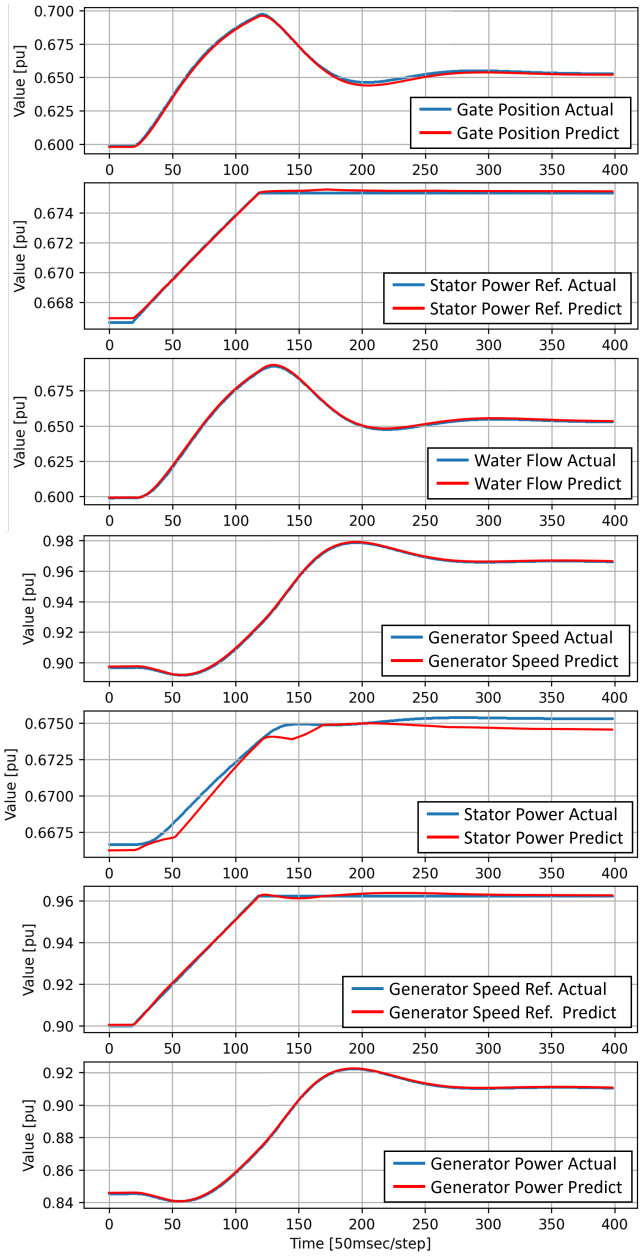

Figure 4: Surrogate Power Plant Output Signals

DQN algorithm, where a neural network is instead used to approximate the Q-values as seen in (4).

$$Q_{(s,a)} = E_{(s,a \to s',r) \sim H} \cdot (r_{(s,a)} + \gamma \cdot max_{a'} Q_{(s',a')}) \quad (4)$$

The neural network policy, E, maps s to Q-values for all possible actions. The loss function is then based on the difference between the predicted Q-values and the target Q-values derived from the Bellman equation. In this case, the Adam optimizer and Huber loss function is used in policy updates. The DDQN algorithm is also an actor-critic method, therefore, there are actually two DQN agents using the same architecture. The actor DQN is the main agent in the environment, while the critic DQN is used to stabilize the training process of the actor DQN (Hessel et al. 2018). The objective of the DDQN agent is to reduce the emulation error of the real-time simulation and physical hydropower plant emulation hardware. Thus, the absolute error (e) between ω_g and ω_s will be monitored and used to train the DDQN. The reward function can be seen in (5).

$$Reward = -error \;\; ; \;\; error = |\omega_g - \omega_s| \quad (5)$$

It should be noted that the hyperparameters of the RL agent may be further optimized, however were chosen not to find the optimal performance, but rather to demonstrate the approach.

Training

The DDQN was trained using outputs of the surrogate hydropower plant being evaluated on validation data sets along with outputs from the surrogate emulation hardware as input observations. The DDQN agent was trained with a learning rate of 0.0001 for 4000 episodes, however, converged around 2000 episodes as seen in Figure 5.

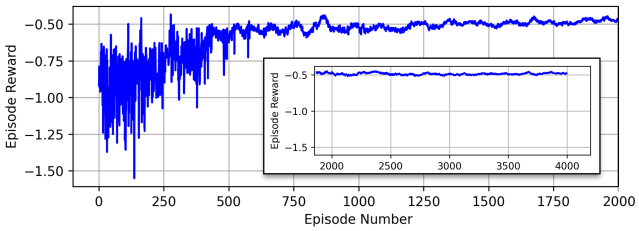

Figure 5: Results from DDQN Agent training

Results

Training results show reduced error in the surrogate model's ability to represent a hydropower plant given a limited data set. Likewise, the RL agent can consistently learn over time from the surrogate environment and perform corrective actions during transient periods of distortion from the surrogate emulation hardware. A single illustrative comparison between the original system without a trained DDQN agent and one with the DDQN agent included can be seen in Figure 6.

Without an agent present, a constant error (Green) is introduced by the surrogate emulation hardware from the original ω_g output coming from the surrogate hydropower plant model (Blue). Once the DDQN agent is introduced, corrective action is taken during the transition period from a power setpoint increase, decreasing the speed error of the emulated shaft. From training, the optimal performance was achieved after 2000 episodes of training. Additional training to 4000 episodes yielded no increase in performance, however, with more tuning and architectural design of the DDQN agent, better performance can be achieved.

The trained RL agent was tested on 7 different generating and pumping events to test the performance during different transient periods. The hydropower events started in steady state and were triggered by adjusting the ω_{ref}. The resulting effect on RL agent reward for different starting ω_{ref} and $\Delta\omega_{ref}$ values can be seen in Figures 7a, and 7b, respectively.

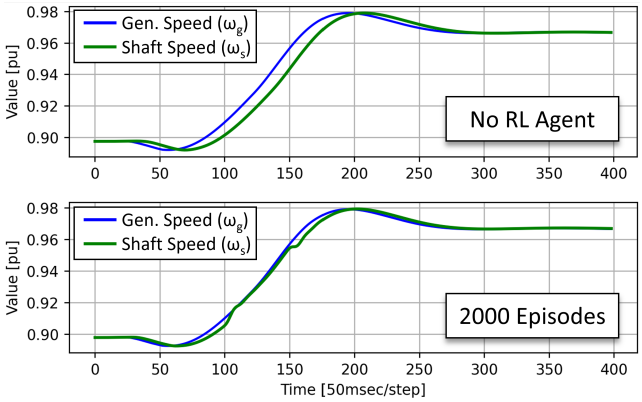

Figure 6: DDQN agent's performance improvement for reducing emulated shaft speed error

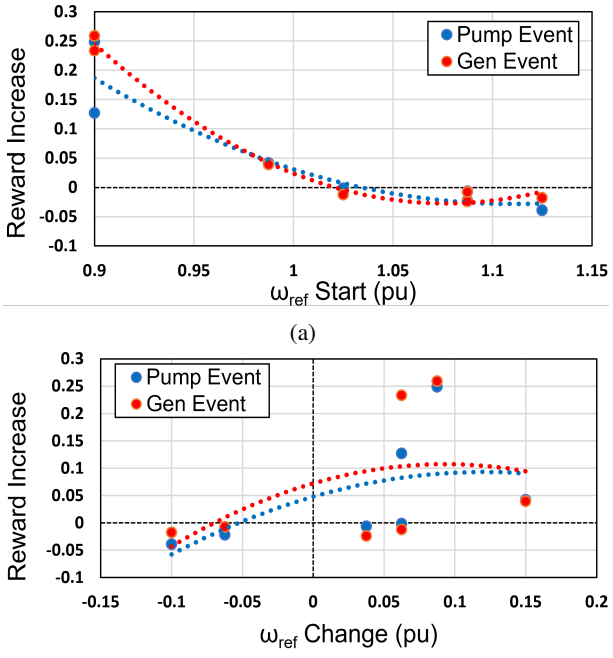

Figure 7: Affect of a) starting ω_{ref}, and b) change in ω_{ref} on emulation performance.

Conclusions and Future Work

In this work, an accurate emulation of hydropower plant dynamics using a physics-based real-time plant model and physical VFD-driven dual IM drive is the final goal. To reduce emulation error and noise from the physical hardware, an RL DDQN agent will be deployed to intercept and modify setpoint signals coming from the real-time simulation model before sending them to the physical hardware. However, training in real-time using live data-streams can produce inefficiencies due to real-time environment constraints leading to poor training. Thus, a surrogate model-based approach to efficient learning of the RL DDQN agent is used.

A hydropower plant is accurately represented using a data-driven ANN with limited data from a physics-based simulation of both generating and pumping modes of operation. The outputs of the surrogate hydropower plant model are provided as inputs to a DDQN agent whose purpose is to reduce error between the generator speed of the hydropower plant model generator and a surrogate emulated hardware shaft. This is accomplished by intercepting and modifying the generator speed signal such that any noise or distortion can be prevented on the output of the surrogate emulation hardware. After training within the surrogate environment, the DDQN agent demonstrates the capability to reduce emulation error. In the future, the pre-trained DDQN will be deployed in a real-time platform and further trained using real-time data streams from the real-time emulation. Eventually the RL algorithm will be deployed in a real-world hardware emulation environment.

Acknowledgments

This work was authored by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided by the U.S. Department of Energy Office of Energy Efficiency and Renewable Energy Water Power Technologies Office through HydroWIRES' project titled 'Emulating Hydropower in a Controlled Real-world Environment at ARIES for Rapid Prototyping of Next-generation Hydro-controls'. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

References

Abid, K. 2022. Digital Twin and Artificial Intelligence Incorporated With Surrogate Modeling for Hybrid and Sustainable Energy Systems. *arXiv preprint arXiv:2210.00073*.

Adam, T. 2022. A comprehensive review of digital twin—part 1: modeling and twinning enabling technologies. *Structural and Multidisciplinary Optimization*, 65(12): 354.

Angione C, Y. E., Silverman E. 2022. Using machine learning as a surrogate model for agent-based simulations. *PLoS ONE 17(2): e0263150.*

Chaturvedi, S.; Bui, V.-H.; Su, W.; and Wang, M. 2023. Reinforcement Learning Based Integrated Control to Improve the Efficiency of DC Microgrids. *IEEE Transactions on Smart Grid*.

Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Hewing, L.; Wabersich, K. P.; Menner, M.; and Zeilinger, M. N. 2020. Learning-Based Model Predictive Control: To-

ward Safe Learning in Control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1): 269–296.

Khalid, A.; Stevenson, A.; and Sarwat, A. I. 2021. Performance analysis of commercial passive balancing battery management system operation using a hardware-in-the-loop testbed. *Energies*.

Kollmer, J. D.; Biswas, S. K.; Bai, L.; Sarwat, A. I.; and Saad, W. 2018. A hardware-in-the-loop experimental platform for power grid security. In *2018 ASEE Annual Conference & Exposition*.

Lazaridis, A.; Fachantidis, A.; and Vlahavas, I. 2020. Deep reinforcement learning: A state-of-the-art walkthrough. *Journal of Artificial Intelligence Research*, 69: 1421–1471.

Li, Y.; Yu, C.; Shahidehpour, M.; Yang, T.; Zeng, Z.; and Chai, T. 2023. Deep Reinforcement Learning for Smart Grid Operations: Algorithms, Applications, and Prospects. *Proceedings of the IEEE*.

Panwar, M. 2022. Data-Driven Approach for Hydropower Plant Controller Prototyping Using Remote Hardware in the Loop (DR-HIL). Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States).

Panwar, M.; Lundstrom, B.; Langston, J.; Suryanarayanan, S.; and Chakraborty, S. 2013. An overview of real time hardware-in-the-loop capabilities in digital simulation for electric microgrids. In *2013 North American Power Symposium (NAPS)*, 1–6.

Poudel, B.; Panwar, M.; and Hovsapian, R. 2023. Data-Driven Scalable Emulation of Hydropower Using Real-Time Hardware-in-the-Loop. Technical report, National Renewable Energy Laboratory (NREL), Golden, CO (United States).

Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378: 686–707.

Song, Y.; Romero, A.; Müller, M.; Koltun, V.; and Scaramuzza, D. 2023. Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. *Science Robotics*, 8(82): eadg1462.

Stevenson, A.; Tariq, M.; and Sarwat, A. 2023. Reduced Operational Inhomogeneities in a Reconfigurable Parallelly-Connected Battery Pack Using DQN Reinforcement Learning Technique. In *2023 IEEE Transportation Electrification Conference I& Expo (ITEC)*, 1–5.