Accelerated DRL Agent for Autonomous Voltage Control Using Asynchronous Advantage Actor-critic

Zhengyuan Xu*§, Yan Zan*, Chunlei Xu†, Jin Li‡, Di Shi*, Zhiwei Wang*, Bei Zhang*, Jiajun Duan*
*GEIRI North America, San Jose, USA
†State Grid Jiangsu Electric Power Company, Nanjing, CN
‡NARI Group Corporation, Nanjing, CN
§University of Pennsylvania, Philadelphia, USA
Email: yan.zan@geirina.net

***Abstract*—This paper presents a novel data-driven parallel framework for autonomous voltage control (AVC) of the power grid. The proposed framework employs a distributed Deep Reinforcement Learning algorithm named Asynchronous Advantage Actor-Critic (A3C) to regulate voltage profiles in a power grid. A well-trained accelerated agent is obtained in the proposed framework by employing multiple workers simultaneously and interacting with a power grid simulator repeatedly. With the proposed framework, multiple threads can run in parallel. A well-trained agent, which utilizes the parameters acquired by the joint training of multiple workers, is obtained and tested through a realistic Illinois 200-bus system with consideration of N-1 contingencies. The training and testing results show the significant speedup capability and excellent numerical stability of the proposed framework.**

***Index Terms*—Artificial Intelligence, Autonomous Voltage Control, Parallel Deep Reinforcement Learning, A3C, On-policy Learning**

I. Introduction

Keeping a secure voltage profile is essential for the power system to prevent cascading outages. With rapidly increased penetration of renewable energy, demand response, and power electronic devices, modern power systems are facing grand challenges in regulating voltage profiles due to the inherently complex dynamics and high uncertainty brought by these elements. Under certain circumstances, local voltage violations could eventually lead to wide-area blackouts if no effective control actions are applied in time [1]. Therefore, maintaining the system's ability to rapidly restore the voltage back to nominal after severe disturbances is of great importance [2].

Traditionally, off-line decision and control strategies are adopted at the device level [3], e.g., manually switching on/off the shunt or adjusting voltage set points of generators. Such control strategies may be only confined to neighboring areas thus have limited control effects. Moreover, the predefined control actions may not be suitable for real-time applications, due to the stochastic nature of the power grids. Several automatic voltage control strategies have been proposed in [4]–[8] to resolve this issue by regulating the voltage profile in a hierarchical manner. However, such control strategy highly relies on the accuracy of the real-time system-wide models. Those models may become inaccurate under large disturbances that violate the quasi-steady state assumptions. Besides, the computation time for deriving optimal control actions may increase exponentially for large-scale power networks, which further limits their applications in real-time.

This work is funded by the SGCC Science and Technology Program.

Recently, Deep Reinforcement Learning (DRL) has made several improvements and achieved state-of-the-art performance on many control and decision-making tasks. These model-free DRL methods do not rely on dynamic system models and can be quickly deployed after training, which makes it suitable for online applications. Besides, once the DRL agent is well trained, it is easily adapted to random variations of system loads and renewable energy resources and provides robust control commands in real-time. Several DRL-based methods have been proposed to regulate voltage in an autonomous manner [8]-[12]. References [9]–[11] reformulate the continuous control problem to discrete control space and employ the discrete control actions, which may not be accurate enough. Reference [12] utilizes an off-policy algorithm, i.e., deep deterministic policy gradient (DDPG), to solve a continuous control problem. Nevertheless, DDPG, as an off-policy learning approach, is reported to have relatively unstable performance and can be sensitive to hyperparameters though it can sample the training experiences efficiently [12].

This paper proposes an accelerated agent using A3C-based approach to solve the autonomous voltage control problem in order to overcome the downside of DDPG-based approaches and aim for the continuous control space. A3C algorithm adopts an on-policy learning strategy, it provides a multi-threaded asynchronous parallel framework, therefore, it is faster, simpler, and more robust comparing to off-policy methods such as DDPG. More specifically, the A3C algorithm has the following advantages. First of all, by asynchronously launching multiple workers, it gathers more training data in unit time, thus expediting the training process. Secondly, it does not rely on experience replay buffer, which is typically used in off-policy methods, and potentially requires less memory and less computation per interaction with the environment. Finally, since every single worker also has its own environment, it receives more diversified data that reduce the correlations between training experiences. Thus the trained agent can be more robust with better overall performance. This paper extends the work presented in [1,11], in which effective DRL agents based on DQN and DDPG were proposed.

The remainder of this paper is organized as follows. Section II provides an overview of the parallel architecture of

the A3C-based accelerated agent and its main components. Section III describes the detailed implementation of the accelerated agent. In Section IV, case studies are conducted with N-1 contingency considered. The agent shows excellent performance for both training and testing stage on the Illinois 200-bus system [13]. Finally, conclusions are reached in Section V, and future research work is also included.

II. The A3C-based Parallel Framework for AVC

A. Parallel DRL Design for AVC

AVC is used to regulate the voltage automatically and keep the voltage profile within a predefined normal range after any disturbances. Conventional methods for AVC are designed highly rely on the accuracy of system-wide models [4]–[8]. DRL is a model-free method that provides an alternative data-driven solution. However, the training process of sequential DRL agent is quite time-consuming and unstable [1], [12]. To tackle this issue, several distributed DRL algorithms have been proposed and reported in [14]–[16]. In this work, we applied one distributed DRL algorithm, i.e., the A3C-based parallel algorithm to build the accelerated agents to provide coordinated, continuous voltage control for power grid operation. The proposed framework supports parallelism in both data usage and model parameters update on a single server with a multi-core CPU. More details regarding each component are provided in the subsequent sections.

B. Overall Architecture Design

The overview architecture of the proposed A3C-based accelerated agent is shown in Fig.1. The accelerated agent utilizes an asynchronous, parallel training pattern to learn more efficiently. With this parallel framework, the training time can be dramatically reduced due to the accelerated convergence rate. Moreover, the performance of accelerated agent on the training set will be more stable compared to the other DRL agents like DQN or DDPG [1], [12].

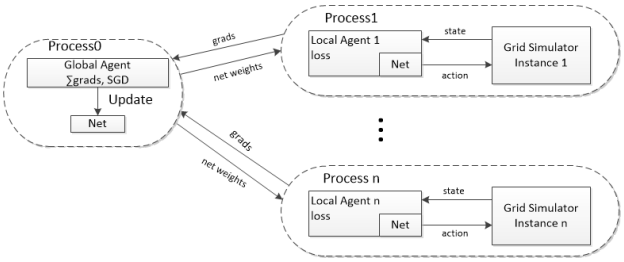

Fig. 1. Diagram of the high-level architecture of the framework.

As can be seen from Fig.1, the parallel framework mainly consists of two components. The central process thread on the left is the master agent, whereas the other children process threads on the right are known as local worker agents. All agents share the same actor and critic neural networks. In the training process, the master agent serves as a central controller to synchronize the shared set of network parameters across different workers. The synchronization takes place by receiving gradient updates computed by local workers asynchronously and then apply changes to the shared networks as soon as they are received. Then the updated neural network parameters need to be broadcasted to all other worker processes, when the worker agents are interacting with multiple Grid Simulator instances using multi-threads. In this way, local worker agents can collect independent experiences from each other, and then calculate the value, the policy losses and the gradients, with respect to its own temporal network parameters.

C. A3C-Based Accelerated Agent

The accelerated agent leverages the A3C-based parallel framework to perform on-policy learning. It can be characterized by three main properties: Asynchronous, Actor-Critic, and Advantage [16].

1) Asynchronous: unlike sequential DQN or DDPG algorithm with a single agent, A3C is a parallel algorithm where multiple worker agents are trained on multiple CPU cores simultaneously on a single server, each with their own copy of the model and the environment.
2) Actor-Critic: it refers to the set of neural networks used by the agents. Each agent is equipped with an actor network and a critic network.
3) Advantage: the advantage is a value that used to replace a normal discounted reward. One benefit of the advantage estimation is that the agent is able to calculate how much better its actions turn out to be comparing to its expectation, allowing the algorithm to concentrate on the difference, and making up to it.

The A3C algorithm learns two networks: the actor (policy) network and the critic (value) network. During the learning process, the actor network adjusts probabilities for actions given the estimated advantage of certain actions. The critic network learns to estimate the advantage based on the feedback from the environment following the actions taken. To further stabilizes the learning process, the A3C algorithm uses an n-step return mechanism, e.g., it updates the policy and value-function every n steps using the n-step return. The policy π and the value function V are updated after t_{max} actions are taken, or when a terminal state has been reached [16].

The gradient update rule for accumulated gradients with respect to policy parameter θ is:

$$d\theta \leftarrow d\theta + \nabla_{\theta'} log\pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v)) \tag{1}$$

and the accumulated gradients with respect to local value network parameter θ_v is updated by:

$$d\theta_v \leftarrow d\theta_v + \partial(R - V(s_i;\theta'_v))^2/\partial\theta'_v \tag{2}$$

where $\pi(a_t|s_t;\theta)$ is the policy and $V(s_t;\theta)$ is the estimation of the value function. $R - V(s_i)$ is the advantage value. Parameters θ' and θ'_v are calculated by accumulating gradients for n steps within each worker thread. The value function for the specific policy π is defined as (the subscript t means the value at time step t):

$$V^{\pi}(s) = \mathbb{E}[R_t|s_t = s] \tag{3}$$

where R_t is the total accumulated discount return from time step t with a discount factor from 0 to 1.

The loss function for the critic (value) network is given by:

$$L_c = (R - V(s_i))^2 \tag{4}$$

And the loss function for the actor (policy) network:

$$L_p = \log(\pi(a_i|s_i) \cdot (R - V(s_i))) - \beta * H(\pi) \quad (5)$$

where $H(\pi)$ is the entropy and β is the normalization factor for the entropy.

To demonstrate how the agent interacts with the environment and learns from the interactions, the steps in one training episode is presented in Algorithm 1.

Algorithm 1: A3C for AVC - pseudocode for each actor-learner thread.

```
Result: Finish when all episodes are trained
Initialize global actor and critic networks parameters randomly;
Initialize t ← 1, T ← 0;
while T < T_max do
    Randomly perform load change (with/without contingency) to obtain initial state S;
    Check for violation or divergence based on S;
    n = 0;
    Reset gradients to 0;
    Synchronize parameters from global network to local network;
    t_start ← t;
    if Violation then
        while terminal s_t or t − t_start == t_max do
            Perform a_t according to policy π;
            Receive reward r_t and new state s_{t+1};
            t ← t + 1;
            T ← T + 1;
        end
        Calculate Reward R;
        Apply discounted Reward to R;
        Calculate gradients and upload to global networks;
    else
        Check for divergence;
    end
end
```

This procedure ensures that each worker thread is communicating with its own instance of the environment, thus to support a parallel training process. This parallelism realizes more server capacity for training, which can substantially accelerate the overall training procedure. More specifically, the more workers to process the training data in parallel, the less time it requires to complete the training. More importantly, the updates of accumulated n-step gradients are sent back to the master by each worker thread in an asynchronous way, which reduces the variance of overall actor-critic model and stabilizes the training process.

III. Deployment of A3C-Based Accelerated Agent

A. The Deployment Platform

In this work, the A3C-based accelerated agent is developed with TensorFlow in Python that runs on a Linux platform with 528 GB memory and 32 cores. The environment used for the training process is an in-house, high-fidelity power grid simulator to mimic the real power system response at the quasi-steady state [1, 11].

B. Components of A3C-based Accelerated Agent

In this section, the following notations are defined in the context specific to the AVC task.

- **Goal** For the proposed accelerated agent, the goal is to generate a set of actions from an infinite, continuous action space when abnormal bus voltages are observed, the agent attempts to regulate the voltage profiles.
- **Episode** As defined in [1], in the AVC task, an episode represents any operating condition collected from real-time measurement systems such as supervisory control and data acquisition (SCADA) or phasor measurement units (PMUs). In the task, episodes are generated by randomly changing the loading conditions, generation dispatches, system typologies, and considering contingencies. N-1 contingencies are randomly selected from 85 lines in this work. Totally 40,000 episodes are partitioned randomly and assigned to multiple workers in order to allow data parallelism.
- **State** The original state in the task is an 890-dimensional vector, including bus voltages, phase angles, active and reactive power flows on transmission lines. To remove less important information for voltage control tasks, we keep the 200-dimensional bus voltage as an input state.
- **Action** Similar to [1, 11], and without loss of generality, we take the generator voltage set points modification as the action to regulate system voltages, and it takes values from a continuous action space within the range of [0.95, 1.05].

The reward function used in the model is defined as follow:

$$R = \begin{cases} \sum_i^N 0.05 - |1 - v_i|, & \forall v_i \in V : 0.95 \leq v_i \leq 1.05 \\ -20, & \exists v_i \in V : v_i < 0.80 | v_i > 1.25 \\ -1000, & \text{if pfs failed with A} \\ -\sum_i^N |1 - v_i| & \text{otherwise} \end{cases} \quad (6)$$

where V is the set of voltages of the 200 buses, and A is the action taken by the agent. The pfs stands for the power flow solver program. The reward of a taken action is determined solely on the voltage value of the buses. As described in equation (6), the exact value of the reward is calculated differently when the voltage of a bus is in four different ranges. Firstly, the reward is positive only when voltages of all 200 buses are within the range of [0.95, 1.05] in $p.u$. If this is the case, then the agent receives the highest reward when the voltage of a bus is at 1.00 pu, and receives a smaller positive reward when the voltage value deviates from the value 1.00 $p.u$. The total reward is the sum of the rewards of all 200 buses. Secondly, a total negative reward is given if any bus voltage drops below 0.80 $p.u$ or rises over 1.25 $p.u$. Finally, when the power flow solver fails to solve the state with the provided action, a hefty penalty is given too. If all buses are within the violation range (0.80, 1.25) but are out of the normal range [0.95, 1.05] in $p.u$, a small negative reward is given for each bus voltage violation. The further away the

voltage is from 1.00 pu, the larger the penalty. Still, in this case, the total reward is the sum of all 200 buses' rewards. The final reward for an episode with n steps is the sum of rewards in each step divided by the number of steps:

$$\text{Final Reward} = \sum_{i=1}^{n} R_i/n \tag{7}$$

The design of this reward function aims to serve the following purposes:

1) Encourages the agent to take actions that drive the voltage of buses towards 1.00 pu as much as possible.
2) Prevents the agent from taking actions that will cause divergence in the power system.
3) Prevents the agent from taking actions that will result in a failed power flow case.
4) Discourages the agent to take actions that lead to abnormal voltage profiles.

C. Neural Network Architecture Design

The same neural network structure is used for all cases in this work. Having a shared neural network architecture indicates that the framework of accelerated agent is robust to hyper-parameter changes i.e., the number of neurons and the number of layers in the networks. For the actor network, three layers of neurons are applied: the first layer consists of a batch normalization layer with 200 neurons and a linear activation function. Then the second layer is a fully-connected layer with 200 neurons and a Rectified Linear Units(ReLU) activation function. The last layer is also a fully connected layer with 38 neurons and a $tanh$ activation function. The critic network has three layers as well. The first layer is a batch normalization layer with 200 neurons and a linear activation function, followed by two fully connected layers, with 100 and 1 neurons, ReLU, and linear activation functions, respectively. The batch normalization layer is proved to be efficient in computing the gradients [17]. It normalizes the layer inputs, and therefore reduces the co-variance shift and increases the stability of a neural network. The optimizer used for both actor and critic networks is the Adam Optimizer with a learning rate of 10^{-4}. We also experimented with other hyper-parameter sets and found that the system is robust to hyper-parameter changes.

IV. Case Studies

In this section, 40,000 cases in total are considered for training the agent. Besides, 10,000 test cases are generated to test the performance of the trained agent. Using a separate test set allows us to evaluate the trained agent's ability to generalize on an unseen dataset.

A. Normal Operating Conditions without Contingencies

In this case, the only perturbation added to the system is the random load variations. Each bus's load is randomly perturbed from 70% to 130% of its original loading condition. After that, the generators are re-dispatched according to their original participation factors, to maintain the active power balance of the system [1]. A total of 40,000 training cases with random load perturbation are generated by the Powerflow & Short circuit Assessment Tool (PSAT), developed by Powertech Labs. In each of the generated cases, information for a converged power flow is provided, including bus voltage, phase angle, and other information representing the operating status of the 200-bus system. Our agent only uses the 200-vector bus voltage to determine its next action. The information for each case is saved in a PTI v33 format. The reward obtained by an A3C-based accelerated agent during the training is given in Fig.2. The figure demonstrates the experiments conducted

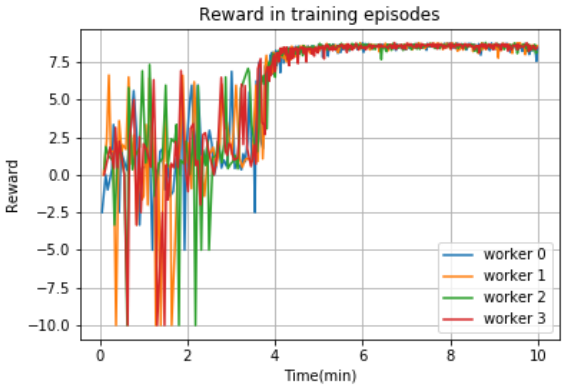

Fig. 2. Reward received in each worker agent in the training process using 40,000 episodes without considering contingencies.

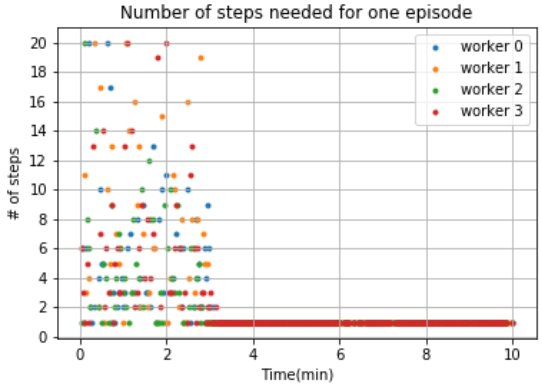

Fig. 3. Number of steps taken for 40,000 training episode without considering contingencies.

with 4 workers. The reward curves show a similar trend for all workers, which is starting with massive fluctuations and ending up with convergence to positive rewards around 8 within 4 minutes. According to the reward function definition in Eq.(6), positive rewards indicate the accelerated agent can provide actions to adjust all abnormal voltages to normal range successfully. Note that the minimum episode reward is -1000, whereas the maximum is around 8, so for visualization purposes, the negative rewards are normalized to -10.

Fig.3 demonstrates the number of steps needed to complete the training process. Note that the maximum number of steps allowed is 20, which indicates if an agent requires more than 20 steps to adjust the voltage, then this episode is considered failed, and a new episode is initiated. The graph shows that four workers demonstrate a common trend for the agent's performance, which explains how the worker is able to reach convergence within four minutes. As the primary control objective, the agent needs to adjust all bus voltages back to the normal range with as few steps as possible. Our experiments show that 10,000 test cases are completed with only one step,

which proves the training is effective.

B. Considering N-1 Contingencies

In this case studies, the N-1 contingency (one line between two buses) is considered to cover the topology change. Unlike the previous case study without contingencies, a topology change introduces more uncertainties to the power grid and therefore is more difficult to handle. The training results are given in Fig.4 and Fig.5. Note that to make the N-1 contingency cases to converge, the agent needs a bit more time than the cases without contingencies, as expected. The testing result on the 10,000 test cases also shows that only one step is needed to eliminate bus voltage violations. In addition, the acceleration from DDPG or DQN to A3C is noticeable. For a DQN agent to converge on the AVC task with N-1 contingency, 12,000 episodes and about 40 minutes of training are required, as described in [1]. For a DDPG agent to converge on the AVC task without contingency, it requires 10,000 training episodes and about 30 minutes of training time, as shown in [12]. However, the accelerated agent trained by four workers in this paper only takes 1,500 episodes and about 4 minutes of training time to converge in both scenarios, demonstrating a considerable acceleration and training speed improvement.

Fig. 4. Reward received in each worker agent in the training process using 40,000 episodes considering N-1 contingencies

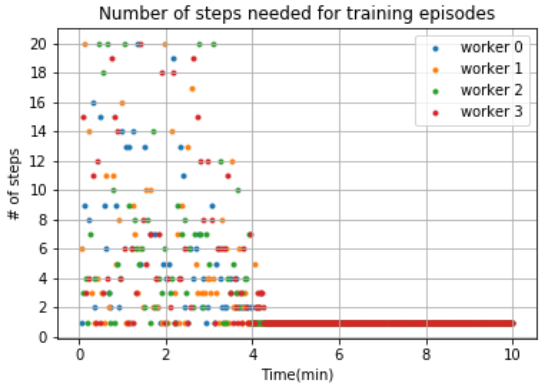

Fig. 5. Number of steps taken for 40,000 test episode considering N-1 contingencies

V. Conclusions

A novel A3C-based parallel framework is presented for training an intelligent accelerated agent for the AVC task. The case studies on the Illinois 200-bus system show that the proposed A3C-based accelerated agent can achieve higher performance on the task compared to a sequential agent. Firstly, it enables data parallelism and gradients to update parallelism in multiple agents, facilitating a faster convergence rate. Secondly, experience replay buffer for DQN/DDPG based agents is no longer needed. Therefore the system can be easily scaled on massive data sets. Finally, the accelerated agent can attain a more diversified training experience as each worker's experience is independent. The independence introduces extra robustness to the training model. As a result, the model is less sensitive to hyper-parameter changes. With all of these features, the A3C-based accelerated agent can behave more stable than other DRL agents for solving AVC problems.

References

[1] R. Diao, Z. Wang, D. Shi *et al.*, "Autonomous voltage control for grid operation using deep reinforcement learning," *IEEE PES General Meeting*, 2019.

[2] J. Duan, H. Xu, W. Liu, J. Peng, and H. Jiang, "Zero-sum game based cooperative control for onboard pulsed power load accommodation," *IEEE Trans. on Ind.l Informat.*, pp. 1–10, 2019.

[3] I. Kamwa, R. Grondin, and Y. Hebert, "Wide-area measurement based stabilizing control of large power systems-a decentralized/hierarchical approach," *IEEE Trans. Power Syst.*, vol. 16, no. 1, pp. 136–153, Feb 2001.

[4] J. P. Paul, J. Y. Leost, and J. M. Tesseron, "Survey of the secondary voltage control in france : Present realization and investigations," *IEEE Trans. Power Syst.*, vol. 2, no. 2, pp. 505–511, May 1987.

[5] S. Corsi, M. Pozzi, C. Sabelli, and A. Serrani, "The coordinated automatic voltage control of the italian transmission grid-part i: reasons of the choice and overview of the consolidated hierarchical system," *IEEE Trans. Power Syst.*, vol. 19, no. 4, pp. 1723–1732, Nov 2004.

[6] S. Corsi, M. Pozzi, M. Sforna, and G. Dell'Olio, "The coordinated automatic voltage control of the italian transmission grid-part ii: control apparatuses and field performance of the consolidated hierarchical system," *IEEE Trans. Power Syst.*, vol. 19, no. 4, pp. 1733–1741, Nov 2004.

[7] H. Sun, Q. Guo *et al.*, "An adaptive zone-division-based automatic voltage control system with applications in china," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1816–1828, May 2013.

[8] ——, "Development and applications of system-wide automatic voltage control system in china," *2009 IEEE Power Energy Society General Meeting*, pp. 1–5, July 2009.

[9] J. G. Vlachogiannis and N. D. Hatziargyriou, "Reinforcement learning for reactive power control," *IEEE Trans. Power Syst.*, vol. 19, no. 3, pp. 1317–1325, Aug 2004.

[10] Y. Xu, W. Zhang, W. Liu, and F. Ferrese, "Multiagent-based reinforcement learning for optimal reactive power dispatch," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1742–1751, Nov 2012.

[11] Y. Chen, S. Huang *et al.*, "Evaluation of reinforcement learning-based false data injection attack to automatic voltage control," *IEEE Transactions on Smart Grid*, vol. 10, no. 2, pp. 2158–2169, March 2019.

[12] J. Duan, D. Shi, R. Diao *et al.*, "Deep-reinforcement-learning-based autonomous voltage control for power grid operations," *IEEE Trans. Power Syst.*, pp. 1–5, 2019.

[13] A. B. Birchfield, T. Xu *et al.*, "Grid structural characteristics as validation criteria for synthetic networks," *IEEE Trans. Power Syst.*, vol. 32, no. 4, pp. 3258–3265, July 2017.

[14] A. Stooke and P. Abbeel, "Accelerated methods for deep reinforcement learning," *arXiv preprint arXiv:1803.02811*, 2018.

[15] A. Nair, P. Srinivasan *et al.*, "Massively parallel methods for deep reinforcement learning," *arXiv preprint arXiv:1507.04296*, 2015.

[16] V. Mnih, A. P. Badia *et al.*, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.