Reinforcement Learning based Intelligent Semiconductor Manufacturing Applied to Laser Annealing

Tejender Singh Rawat, Chang-Yuan Chung, Shih-Wei Chen, and Albert Lin

Abstract- **The recent shift in the paradigm of the industrial revolution, i.e., Industry 4.0, has forced industries to reemphasize manufacturing standards to improve the manufacturing and production systems. In this work, instead of more popular supervised and unsupervised learning, we employ reinforcement learning (RL) to determine the process parameters to minimize the sheet resistance of the device. We have used seven independent variables used in fabricating the devices such as dopant ions (*I*), dose (*D*), energy (*E*), wavelength (*W*), repetition rate (*Rep*), temperature (*T*), and power (*P*). The trained RL agent will identify the combinations of independent variables parameters that minimize the sheet resistance. Advantage actor-critic (A2C) and deep Q-network (DQN) agents have been used with batch sizes of 2, 5, and 10, with exploration rates of 0 and 0.2 for every batch size. Both agents with a batch size of 2 and an exploration rate of 0.2 have given the best result. We also show that compared to the Taguchi method, RL can potentially provide faster locating of optimal experimental conditions.**

Keywords: Reinforcement learning, intelligent manufacturing, semiconductor process, laser annealing, machine learning.

I. INTRODUCTION

In the new age of industrial revolution, i.e., Industry 4.0, industries have developed strategic guidelines to inherit the emerging technologies such as big data analytics, cyber-physical system (CPS), data science, cloud computing, and internet of things (IoT) to transform the manufacturing process [1, 2]. Industry 4.0's concept of intelligent manufacturing (IM), refers to the application of artificial intelligence (AI), IoT, and sensors in manufacturing to make intelligent decisions through real-time communication [3]. Industry 4.0 mainly aims at adopting new technologies and methodologies for better-optimized design, manufacturing quality, and production in modern factories [4].

With the integration of these high throughput technologies with sensors, IoT has resulted in the multifold increase of manufacturing data of high dimensionality. In such scenarios, machine learning (ML) and deep learning (DL) algorithms can help to develop strategies to analyze, diagnose and predict the patterns from high dimensionality data automatically. ML tools such as supervised learning (SL) and unsupervised learning (USL) have been extensively used in manufacturing industries for process monitoring, control, optimization, fault detection, and prediction [5-12]. Even in the semiconductor industry, where the processes are complex and fragile, ML provides tools for continuous quality improvement [7, 13-15]. However, SL/USL have certain challenges, such as that they are usually limited to the specific process of the entire manufacturing system [16]. Alternate to SL/USL, reinforcement learning (RL) is another ML approach where no supervision is mandatory for training the model [17]. RL is a well-known approach to be used when we want our model to learn on its own to make decisions and has been used in many optimization problems [18-20].

Specifically, in manufacturing industries, the production environment is often dynamic and non-deterministic and sometimes has to deal with unexpected scenarios or incidents [21]. In such a stochastic environment where the randomness in the dataset makes prediction difficult, RL is more capable of learning the process instead of SL/USL methods [22-28]. In the semiconductor industry, RL has been used in scheduling and dispatching in literature. In particular, Washneck et al. used deep RL for production scheduling for optimization and decentralized self-learning [29], Stricker et al. presented deep RL for semiconductor dispatching, and improved system performance has been demonstrated [30]. In general, RL in the semiconductor industry has been used extensively for production control [31] or scheduling [29, 32-35]. There have been fewer works using RL in intelligent semiconductor manufacturing, especially compared to the efforts in SL/USL. There have been some semiconductor process-control efforts using RL at the theoretical side [36-38]. Khader et al. have used RL in surface mount technology (SMT) in printed circuit board (PCB) with experimental dataset [39].

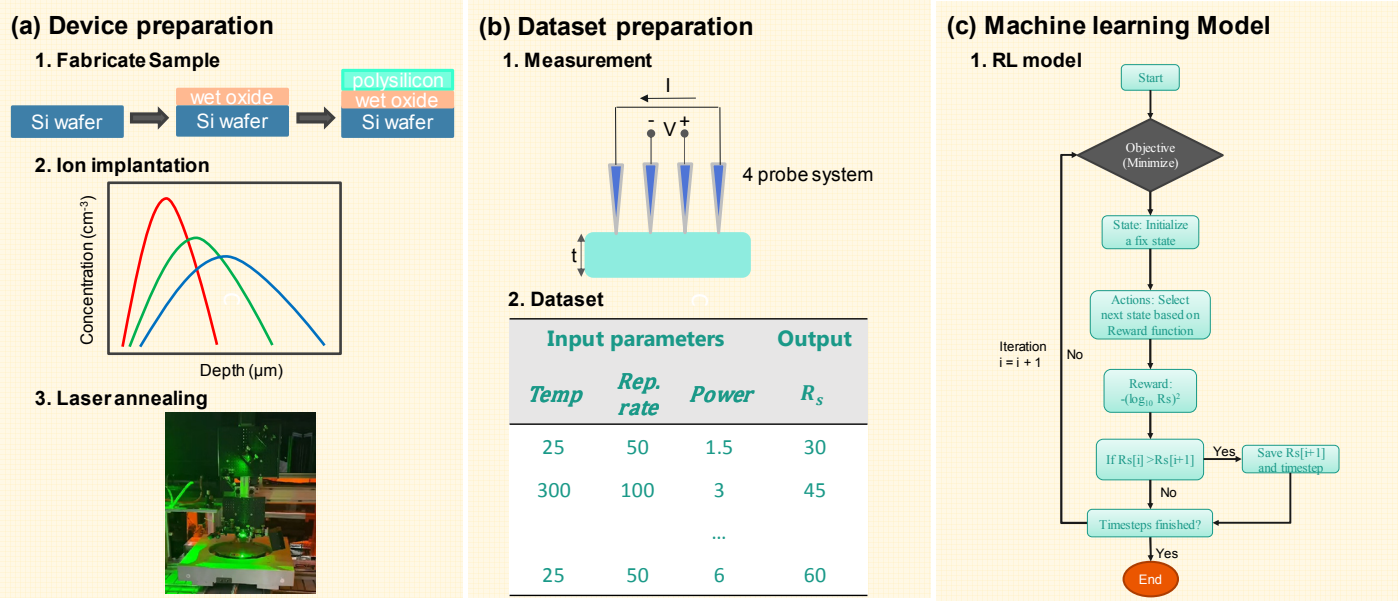

Fig. 1. Summary of the experimental setup for RL optimization. (a) Various steps are involved in device preparations, such as deposition of wet-oxide and polysilicon, ion-implantation and laser annealing, (b) device measurement using 4 probe system for sheet resistance and dataset, and (c) RL model implemented for this work.

The advantage of using RL in the semiconductor industry is that it has the ability to observe complex and dynamic problems and can generalize the strategy for a solution with efficient data utilization [22, 40]. Furthermore, RL has the advantage of splitting the major task into several subtasks, which results in a flexible decentralized structure, and hence reduced computation time is achieved through parallelization [41]. In our previous attempt, we used the RL agent to analyze its performance against the experienced human knowledge in the semiconductor fabrication field [42]. In this work, RL agents will be trained for optimizing the process parameters for minimizing the sheet resistance of the laser-annealed samples.

In this work, the silicon wafers were doped with impurities using the ion-implantation method. Then, instead of using the traditional annealing process to restore the wafer's disordered crystallinity to adjust the electrical conductivity after doping, we used the laser annealing method. The laser annealing process was chosen to avoid the undesired consequences of annealing the entire wafer [43]. Then, the RL-based AI agent will be trained to minimize the sheet resistance, R_s, of laser annealed silicon wafers. The trained AI agent then can take the sequence of actions after analyzing the state of the model with specified limitations. To train the RL agent, the environment will have the following three main sections. First, states section where an agent will observe the change in wafer's sheet resistance based on various variable parameters such as the wavelength of the laser, power, dose, ion, energy, repetition rate, and temperature. This will provide the observation space for the agent. Second, the action of the agent will change the levels of the variables according to the policy. Third, continuous evaluation of the agent's action will be carried out by providing numerical rewards as feedback.

II. METHODOLOGY

A. Sample Fabrication and Dataset Preparation

First, 6-inch p-type silicon wafers of resistivity 1~10 Ω.cm were cleaned using the STD process. In STD clean process, wafers were cleaned using a wet bench before high-temperature deposition on wafers. First, the wafers ware cleaned by in a solution of $NH_4OH:H_2O_2:H_2O$ in the ratio of 1:4:20 at 75°C for 10 minutes. The wafers were rinsed after the SC1 process, then the wafers are again cleaned using the SC2 process which includes the solution of $HCl:H_2O_2:H_2O$ in the ratio of 1:1:6 at 75°C for 10 minutes. Again after rinsing, the wafers were cleaned with diluted hydrofluoric acid (DHF) which includes the $HF:H_2O$ in the ratio of 1:50 at room temperature for 1 minute. After the DHF process, wafers were rinsed and dry spun.

Then, SVCS horizontal furnace system was used to deposit wet oxide and polysilicon of thickness 5000Å and 1000Å, respectively, on silicon wafers. Ellipsometer M2000 was used for measuring the thickness of wet oxide and polysilicon deposition on the wafer. The detailed experimental setup for RL optimization is shown in Fig. 1.

Then, doping of arsenic (*As*) and phosphorous (*P*) was done on the wafers using the ion-implantation. Varian E-500HP implanter machine was used for this process with 10 keV, 25 keV, 40 keV, and 55 keV of ion energy. For every ion energy, the dose of 5×10^{14}, 8×10^{14}, 2×10^{15}, and 5×10^{15} cm^{-2} is used. After implantation, wafers were sliced into a 1.7×1.7 cm^2 shape, and then these samples were laser annealed. Spectra physics HIPPO laser was used for annealing. Nd:YAG laser with wavelengths of 355nm and 532nm with an output power of 0.5-3 W and 1.5-6 W, respectively, were used in the annealing process. The process parameters for the whole annealing process are shown in Table I.

Metal 4-probe from NAPSON CORP. is used for measuring the sheet resistance of samples size 1.7×1.7 cm^2 after the annealing process. The sheet resistance of the samples was measured before and after the annealing process. CAMECA IMS 7F double-focusing mass spectrometer is used for analyzing the depth profile of the dopants.

Table I. Process parameters of laser annealing.

Wavelength (nm)	Repetition Rate (kHz)	Power (W)	Temperature (°C)
532	50	1.5	25
			300
	100		25
			300
	50	3	25
			300
	100		25
			300
	50	4.5	25
			300
	100		25
			300
	50	6	25
			300
	100		25
			300
355	50	1	25
			300
	100	0.5	25
			300
	50	1.66	25
			300
	100	1	25
			300
	50	2.33	25
			300
	100	1.5	25
			300
	50	3	25
			300
	100	2	25
			300

B. Optimization Algorithm

In this work, we have implemented our model using the Tensorforce 0.6.5 library for RL [44, 45], Tensorflow 2.7.0 [46], Python 3.8.13 [47], Numpy 1.21.2 [48], Scipy 1.8.1 [49], and Pandas 1.4.2 [50]. We have considered Taguchi orthogonal array [51] as a baseline model for comparison.

1) RL Method

In RL, an agent learns the situation described by its environment using many trails by taking various action sequences. An agent takes an action $a_t \in A$ based on the state $s_t \in S$ observed at the time t while interacting with its environment in a closed loop, where A and S are all possible actions and states available to the system. The environment then rewards the agent based on the action and presents a new state (s_{t+1}) for an agent to act. This whole sequence is repeated to maximize the total rewards from the environment. This discrete and random process of states and actions makes the process a Markov decision process (MDP). In MDP, mapping from states to action is trained, leading to an optimal policy π^*. The procedure can be formulated as [52, 53].

$$\pi^* = \arg\max_{\pi} \mathrm{E}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)] \quad (1)$$

where T is the length of the episode, R is the reward function at time t and $\gamma \in (0, 1)$ is the discount factor. Hence, in RL, the learning of the agent is usually taken care of by the reward system, as it makes sure that the agent is taking the actions in the right direction and not swaying from its main goal. RL emulates the human learning process by sequentially interacting with the environment.

2) Taguchi Orthogonal array

The Taguchi method [51] is an effective method to observe the effects of various parameters on a performance of a process. This method involves using orthogonal arrays in optimizing the parameters of experiments to determine the optimal input parameters. This method is effective in determining the contribution of significant variables out of many independent variables. This method maintains the robustness of the experiment against all environmental conditions and also helps in minimizing the variation around a target value.

III. RL BASED OPTIMIZATION

In this work, samples were segregated after ion implantation. Every sample of a particular repetition rate and power was annealed either at room temperature or at high temperature, as shown in Table I. Then, every sample was measured for its sheet resistance before and after the annealing process. The whole fabrication process variables serve as states of an environment.

In the context of optimization, the goal of the RL model is to minimize the expected sheet resistance of the sample with respect to the available process parameters. The optimization problem is implemented with Deep Q-Network (DQN), and Advantage Actor-Critic (A2C) agents. DQN is a value-based model-free agent. Value-based RL agent focuses on finding the policy by estimating the value function *V(s)* and action-value function $Q(s, a)$. A2C agent, on the other hand, is considered to be both value-based as well as the policy-based agent. In A2C agent, the value function, *V(s)* is represented by the critic model, whereas the policy function by the actor model. The policy and value functions are typically used by both actor and critic models. This enables an approximate estimation of the learnable parameters for the subsequent training of the model because the critic model offers the measure of the action taken by the actor model.

A. Preparation of semiconductor device for RL

In sample preparation, we used the ion-implantation method to dope the polysilicon layer. The range, R, is the distance of the dopants from the surface of the layer until it comes to rest. In reality, however, this depth can vary and depends on the actual distance an ion travels and hence referred to as the mean projected range, R_p [54].

$$R_p = \frac{1}{N}\sum_i x_i \tag{2}$$

where N is concentration and x_i is the projected range of i^{th} ion. R_p changes with ion energy. The actual depths of individual ions distribute over depth and can be modeled by a Gaussian distribution, with standard deviation or straggling, ΔR_p, expressed as [54].

$$\Delta R_p = \left[\frac{1}{N}\left(\sum_i x_i^2\right) - R_p^2\right]^{\frac{1}{2}} \tag{3}$$

The ion depth distribution with parameters R_p and ΔR_p [54] is expressed as:

$$N(x) = N_{\max} e^{-(x-R_p)^2/2\Delta R_p^2} \tag{4}$$

$$N_{\max} \simeq \frac{0.4\phi}{R_p} \tag{5}$$

where *N(x)* is the concentration of the implanted ions, $N_{\max}$ is the peak concentration, and Φ is the dose concentration. To activate the dopants and restore the structure of the semiconductor, the laser-annealing method is used to avoid the diffusion of the dopants deeper into the structure. The sheet resistance, R_s, of the device can be defined as a product of the integral relation between $\mu(x)$, which is concentration-related mobility, and impurity concentration, *N(x)* [55]:

$$R_s = \frac{1}{q\int_0^{x_i} \mu(x)N(x)dx} \tag{6}$$

where q is the electronic charge of magnitude 1.6×10^{-19} C.

B. RL model

The problem of minimizing the sheet resistance and mapping input parameters according to the resistance using the RL model is formulated using the Tensorforce library for RL. This problem of minimizing can be formulated as an MDP problem and can be

represented as M = (*S, A, P, R*), where *S, A, P* and *R* denote state, actions, probability between two states, and reward, respectively. In accordance with the minimization problem, the purpose of our model is to learn the course of actions that minimizes the sheet resistance value. To lay down the parameters for RL using the Tensorforce library, we have to explicitly define the parameters of MDP in Tensorforce environment as follows:

State-space: The state space, *S*, consists of the information of 7 variables such as ion (*I*), dose (*D*), energy (*E*), wavelength (*W*), repetition rate (*Rep*), temperature (*T*), and power (*P*), and is defined as follows: $S = (s = (I, D, E, W, Rep, P, T))$. We have arranged the dataset according to the state space, *S*. During experiments, slight variations in the power and temperature values can exist, we have fixed the values of power and temperature in RL, as shown in Table I.

Action space: The action space, $A = (a_i = (a_1, a_2, \ldots, a_N))$, consists of all actions a_i, where *i* denotes the action to assign to the i^{th} variable defined in the state space, and *N* is the number of variables. For every variable, each action in a_i consists of integer values. For example, for the ion variable, we used two dopants, i.e., *As* and *P*. Therefore, an agent will have only two actions for the ion variable to choose. Similarly, for every variable, an agent will have respective action to select.

Reward: The agent will decide to converge towards the minimum resistance value according to the defined reward function. Thus, the reward function $R(s_t, a_t)$ for a specific state and action at timestep *t* is defined as follows: $R(s_t, a_t) = -(R_s)$, where an agent will try to maximize its reward value by selecting the minimum value of the sheet resistance.

IV. RESULTS AND DISCUSSION

The cross-sectional SEM images of the fabricated samples are shown in Fig. 2 where the thicknesses of deposited layers are labeled. SEM images confirm the deposited thicknesses of 100nm polysilicon and 500nm wet-oxide over the p-type silicon wafer. Fig. 2(a) and (b) have arsenic as a dopant with a concentration of 5×10^{15} cm^{-2} and energy of 25keV and 55keV, respectively. Fig. 2(c) and (d) have phosphorous as dopant with a concentration of 5×10^{15} cm^{-2} and energy of 25keV and 55keV, respectively.

Fig. 3 shows the images from secondary ion-mass spectrometry (SIMS). SIMS is a very useful tool for analyzing the surface composition and studying the depth profiling of the dopants in the multi-layer sample. SIMS provides a detailed analysis of the composition formed on the sample. For depth profiling, SIMS is very efficient in determining even the very low concentration of sub-ppm or ppb of dopants. In Fig. 3, the SIMS profile confirms the presence of *As* dopant and its diffusion along with the depth of the samples, for annealed and non-annealed samples. Samples in Fig. 3 were annealed at room temperature using a green light laser with a repetition rate of 50kHz and two different powers of 5.999 and 2.997W, respectively. SIMS profile also confirms the diffusion of *As* dopant with respect to the power used in the annealing

A. Environment Setup

For our agent to act and choose the input variables for minimal sheet resistance, we have defined our dataset as the environment. The objective function of the environment is to provide a reward value. These state values will be decided by the action of the agents. The action of our agent is to change the values of one or more variables for every time step according to the policy. For training, the agent will take an action to change the levels of state values: $s = (I, D, E, W, Rep, P, T)$. For an agent to start the training, we provided the initial values of state space as follows: $s = (P, 2\times10^{15}, 40, 532, 100, 1.5, 25)$. Hence, an agent in the RL framework continuously interacts with its environment to learn the sequence of actions it needs to take in any given environment state to maximize its reward.

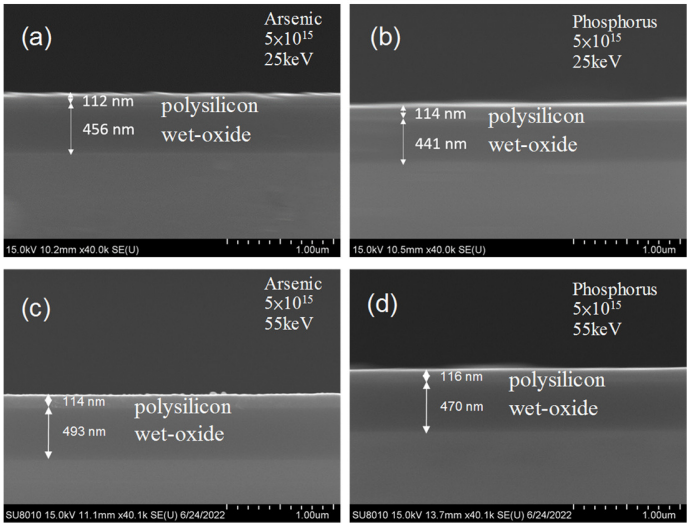

Fig. 2. Cross-sectional SEM images of the samples. Arsenic as dopant with concentration 5×10^{15} cm^{-2} and energy of (a) 25keV and (b) 55keV, respectively, and phosphorous as dopant with concentration 5×10^{15} cm^{-2} and energy of (c) 25keV and (d) 55keV, respectively.

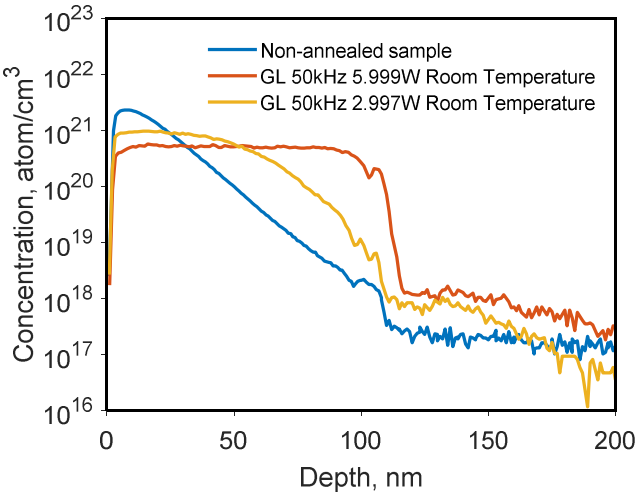

Fig. 3. SIMS profile confirms the presence of arsenic dopant in the samples. Dopant profiles shown here for the annealed sample were annealed in green light laser at a 50kHz repetition rate and power of 5.999 and 2.997W.

B. Training and Result

For training, we have opted for A2C and DQN RL agents. Both agents were trained for 500 episodes. The network for the agent was kept at auto with a layer size of 100, a depth of 2 and 5 layers, and a final depth of 1 layer for both agents. The batch size determines the update frequency at the state action reward table. Frequent update leads to faster convergence, while less frequent update leads to an improved exploration of the sample space. The agent's exploration was varied to either 0.0 or 0.2 for a batch size of 2, 5, and 10, respectively, as shown in Table II. The exploration parameter is a critical parameter of an RL agent, which defines the probability of randomly selecting the next state during RL. This parameter defines the agent's ability to explore the environment rather than be confined to limited space. This helps in avoiding the agent's convergence towards the local minima. As shown in Table II, the effect of the exploration rate among the agents is pronounced. A2C agents with an exploration rate of 0.2 have explored more values of resistance as compared to the A2C agent.

In RL, exploration and exploitation is still very complex and open area. The exploration rate allows the agent to explore more actions in a rewarding way to reach the goal. Without exploration rate, an agent can get stuck in local optima. However, if the current policy explores more than required, the agent will learn nothing. On the contrary, if the agent exploits overly, it will learn actions that are not optimal. For proper training, we want our agent to explore, as well as exploit the actions which can give maximum rewards. As shown in Table II, for the exploration rate of 0, both the agents are stuck in the local optima and were unable to converge towards the minimum value. Whereas, for an exploration rate of 0.2, the performance of both the agents has improved and was more successful in converging towards the minimum value.

Combining a low exploration rate of 0.0 and 0.2 with a low batch size has allowed our agent to explore enough to find the target. As shown in Table II, both A2C and DQN agents have explored ~40-50 different values of sheet resistance for every batch size. The best time step parameter of Table II shows the time step of the agent where it found the lowest value of sheet resistance. This value can be used to compare to other optimization algorithms for algorithm effectiveness. In semiconductor manufacturing, the number of experimental trials corresponds to cost, and thus this number is influential.

In Fig. 4, reward function plots for A2C and DQN agents with an exploration rate of 0 and 0.2, batch size of 2, and network with 2 and 5 hidden layers have been shown. Fig. 4(a) and (b) shows the A2C agent's reward in the training for the first 50 timesteps with a network of 2 layers and batch size of 2 and an exploration rate of 0 and 0.2, respectively. Fig. 4(c) and (d) shows the DQN agent's reward in the training for the first 50 timesteps with a network of 2 layers and batch size of 2 and an exploration rate of 0 and 0.2, respectively. Fig. 4(e) and (f) shows the DQN agent's reward in the training for the first 50 timesteps with a network of 5 layers and batch size of 2 and exploration rate of 0 and 0.2, respectively. It can be seen from Fig. 4 and Table II that when the exploration rate is 0.2, smaller sheet resistance values can be found in most of the RL runs. On the other hand, at a zero exploration rate, the effectiveness of RL is reduced. It is also observed that the

Table II. Hyper-parameter values were used for A2C and DQN agents along with the results obtained in the first 50 time steps in training for each variation of parameters.

Agent	Network	Reward	Batch Size	Exploration	No. of R_s values explored	Result	Best Timestep

					by agent		
A2C	size=100, depth=2, final_depth=1	$-R_s$	2	0 / 0.2	46 / 48	96.483 / 54.247	17 / 48
A2C	size=100, depth=2, final_depth=1	$-R_s$	5	0 / 0.2	46 / 49	104.4 / 50.58	20 / 48
A2C	size=100, depth=2, final_depth=1	$-R_s$	10	0 / 0.2	46 / 48	104.4 / 50.58	20 / 48
A2C	size=100, depth=5, final_depth=1	$-R_s$	2	0 / 0.2	46 / 46	47.4 / 44.323	46 / 47
A2C	size=100, depth=5, final_depth=1	$-R_s$	5	0 / 0.2	46 / 49	104.4 / 50.58	20 / 48
A2C	size=100, depth=5, final_depth=1	$-R_s$	10	0 / 0.2	46 / 48	104.4 / 50.58	20 / 48
A2C	size=100, depth=2, final_depth=1	$-(\log_{10} R_s)^2$	2	0 / 0.2	44 / 49	94.483 / 54.247	17 / 48
A2C	size=100, depth=2, final_depth=1	$-(\log_{10} R_s)^2$	5	0 / 0.2	47 / 49	96.483 / 50.58	17 / 48
A2C	size=100, depth=2, final_depth=1	$-(\log_{10} R_s)^2$	10	0 / 0.2	46 / 48	104.4 / 50.58	20 / 48
A2C	size=100, depth=5, final_depth=1	$-(\log_{10} R_s)^2$	2	0 / 0.2	45 / 43	48.4 /47.54	46 / 40
A2C	size=100, depth=5, final_depth=1	$-(\log_{10} R_s)^2$	5	0 / 0.2	47 / 49	96.483 / 54.247	46 / 48
A2C	size=100, depth=5, final_depth=1	$-(\log_{10} R_s)^2$	10	0 / 0.2	46 / 48	104.4 / 50.58	20 / 48
DQN	size=100, depth=2, final_depth=1	$-R_s$	2	0 / 0.2	38 / 43	48.4 /48.4	37 / 47
DQN	size=100, depth=2, final_depth=1	$-R_s$	5	0 / 0.2	31 / 43	52.333 / 63.37	38 / 15
DQN	size=100, depth=2, final_depth=1	$-R_s$	10	0 / 0.2	35 / 43	54.887 / 57.45	17 / 13
DQN	size=100, depth=5, final_depth=1	$-R_s$	2	0 / 0.2	15 / 40	64.15 / 51.35	13 / 32
DQN	size=100, depth=5, final_depth=1	$-R_s$	5	0 / 0.2	16 / 36	57.987 / 51.35	11 / 32
DQN	size=100, depth=5, final_depth=1	$-R_s$	10	0 / 0.2	14 / 34	48.61 /48.4	5 / 35
DQN	size=100, depth=2, final_depth=1	$-(\log_{10} R_s)^2$	2	0 / 0.2	32 / 42	52.333 / 62.24	22 / 48
DQN	size=100, depth=2, final_depth=1	$-(\log_{10} R_s)^2$	5	0 / 0.2	28 / 42	78.997 / 52.333	14 / 32
DQN	size=100, depth=2, final_depth=1	$-(\log_{10} R_s)^2$	10	0 / 0.2	29 / 43	71.507 / 47.54	18 / 9
DQN	size=100, depth=5, final_depth=1	$-(\log_{10} R_s)^2$	2	0 / 0.2	21 / 39	57.987 / 44.323	11 / 47
DQN	size=100, depth=5, final_depth=1	$-(\log_{10} R_s)^2$	5	0 / 0.2	20 / 37	57.45 / 51.35	11 / 32
DQN	size=100, depth=5, final_depth=1	$-(\log_{10} R_s)^2$	10	0 / 0.2	18 / 39	48.61 / 45.19	5 / 31

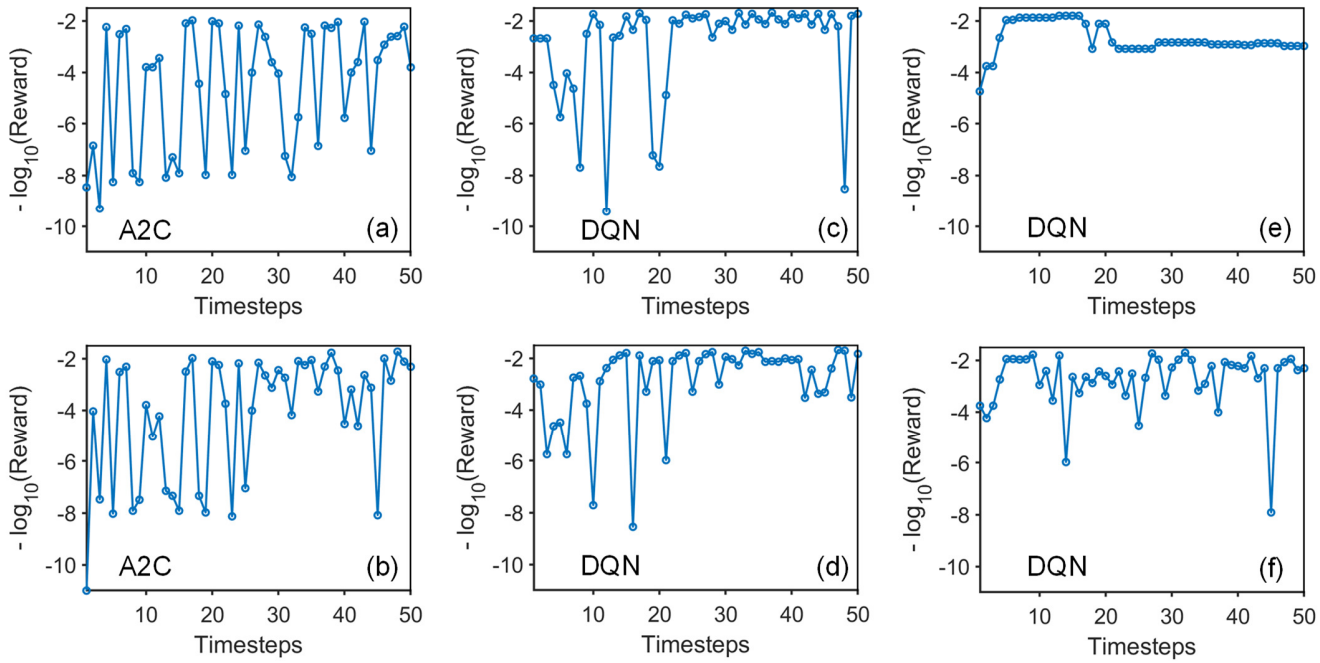

Fig. 4. Reward vs. time steps for the A2C and DQN agents. Reward vs. time steps using A2C for (a) and (b) the first 50 time steps of training using 2 hidden layers of the network, batch size of 2 and an exploration rate of 0 and 0.2, respectively. Reward vs. time steps using DQN for (c) and (d) the first 50 time steps of training using 2 hidden layers of the network, batch size of 2 and an exploration rate of 0 and 0.2, respectively. Reward vs. time steps using DQN for (e) and (f) the first 50 time steps of training using 5 hidden layers of the network, batch size of 2 and an exploration rate of 0 and 0.2, respectively.

minimum resistance value may not be the last step during the 50-time-step RL runs using either DQN or A2C agent due to 50 time step is not a large number for constructing a complete state-action-reward table. Nonetheless, in terms of optimization, fewer trials, i.e., time steps, are desired, and therefore RL should not be executed for an extended time. If RL is trained for thousands of time steps in our study case, the state corresponding to the minimum resistance values will mostly be achieved at the end of training and the evaluation afterward, though thousands of time steps are regarded as ineffective optimization.

Table III. The result of the Taguchi array.

Type of Taguchi Array	No. of R_s values explored in each subspace	Sheet Resistance
2 level	**8**	**113.92(avg)**
3 level	**27**	**83.90 (avg)**
4 level	**64**	**57.99**

Table III shows the results obtained using the Taguchi method for the design of the experiment (DOE). Instead of a full factorial array, Taguchi's method utilizes partial factorial orthogonal arrays where the priority is to avoid the high cost, including the time and cost to run the experiments. Here, the Taguchi method with 2, 3, 4 levels, and 7 variables are implemented [51]. While in our dataset, some variables are of 2 levels and some are of 4 levels, combination and expansion similar to convention practice in employing Taguchi are conducted before locating optimum. Since there are different ways to combine and extend the levels in Taguchi method in order to fit the experiment, averaged values for 2-level and 3-level cases are shown in Table III. The average is taken on 8 and 128 combinations and expansion cases for 2- and 3-level Taguchi methods. Table III shows optimization results using the Taguchi method. The mean values of sheet resistance are shown for 2, 3, and 4 levels of the input experimental parameters. Here, the mean values achieved are 113.92, 83.90, and 57.99 using 2, 3, and 4 levels, respectively.

In comparison to Taguchi's method, both RL agents were more successful in determining the minimum value of 43.323 of sheet resistance. Both A2C and DQN agents have used the same network size (100 neurons, depth=5, final_depth=1), batch size, and exploration rate of 0.2, to find the minimum value of sheet resistance. Both agents have taken the same timesteps of 47 to find the minimum value. The value of minimal sheet resistance visited by A2C and DQN agents is 43.323, and the number of R_s values required to locate the minimum is 46 and 39, respectively. In comparison, Taguchi's array uses 64 data points to locate a minimum of 57.99, which is regarded as a worse performance reference to RL.

There can be many aspects that affect the implementation of RL in the semiconductor manufacturing industry. The first and most important is to design an efficient reward function for an agent. Currently, in this work, we use the sheet resistance values as the reward. Alternatively, there can be other ways to define the reward. For example, a uniform reward of +1/-1 to indicate the decrease or increase in the objective function. Random reward, in some cases, can also lead to improved performance, though the concept is similar to exploration. Thirdly, as compared to SL and USL, instead of using a fixed dataset, RL trains the agent based on the data collected through the interactions with the environment, which can save the trial-and-error time during process condition optimization. In SL and USL, the prediction accuracy highly depends on if the data collection properly spans the entire sample space. If the SL is going to be used to optimize the process condition, the data collection should be extended over all possible choices in process parameters. This leads to inefficient optimization. On the other hand, RL is a Markov decision procession, and the data needed is collected in the run, eliminating redundant data collection steps.

There can be some aspects that can improve the RL's performance, especially in semiconductor manufacturing. First, instead of model-free models, model-based learning makes it easier to learn the solutions to a problem [56, 57]. Additionally, model-based learning requires fewer samples as compared to model-free methods. Second, RL can be used as a fine-tuner on some priors, such as AlphaGo, this project was initiated as supervised learning, and then RL was used on top for fine-tuning. Similarly, transfer learning can provide the priors which can leverage the knowledge from the previous tasks [58, 59]. Moreover, RL can also be coupled with global optimization methods for optimization problems [60, 61].

V. CONCLUSIONS

This work implements the RL in laser annealing in semiconductor manufacturing. We have implemented RL using the tensorforce library on the samples with various independent process parameters. The objective of this work is to find the parameters corresponding to the minimum sheet resistance, Rs. For RL, the information of 7 independent variables such as ion (*I*), dose (*D*), energy (*E*), wavelength (*W*), repetition rate (*Rep*), temperature (*T*), and power (*P*) will be the state space, *S*. Agent action, *A*, will be to select the next state based on reward function. A2C and DQN agents with exploration rates of 0 and 0.2, network size of 100 neurons with two hidden layers, have been implemented for the task. Compared to Taguchi's method, 43.323 of sheet resistance is located by A2C and DQN in RL using 46 and 39 samples, respectively, which is lower than the 64 samples by Taguchi's orthogonal arrays. We believe RL has large potential in intelligent manufacturing in terms of reduced cost and shortened trial-and-error cycles.

REFERENCES

[1] T. Kotsiopoulos, P. Sarigiannidis, D. Ioannidis, and D. Tzovaras, "Machine Learning and Deep Learning in smart manufacturing: The Smart Grid paradigm," *Computer Science Review,* vol. 40, p. 100341, 2021/05/01/ 2021.

[2] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *Journal of Manufacturing Systems,* vol. 48, pp. 144-156, 2018/07/01/ 2018.

[3] X. Yao, J. Zhou, J. Zhang, and C. R. Boër, "From Intelligent Manufacturing to Smart Manufacturing for Industry 4.0 Driven by Next Generation Artificial Intelligence and Further On," in *2017 5th International Conference on Enterprise Systems (ES)*, 2017, pp. 311-318.

[4] A. Kusiak, *Intelligent manufacturing systems*. Englewood Cliffs, N.J.: Prentice Hall, 1990.

[5] E. Alpaydin, *Introduction to Machine Learning*: The MIT Press, 2010.

[6] R. Gardner and J. Bicker, "Using Machine Learning to solve tough manufacturing problems," *International Journal of Industrial Engineering,* vol. 7, pp. 359-364, 12/01 2000.

[7] D. T. Pham and A. A. Afify, "Machine-learning techniques and their applications in manufacturing," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture,* vol. 219, pp. 395-412, 2005/05/01 2005.

[8] U. Çaydaş and S. Ekici, "Support vector machines models for surface roughness prediction in CNC turning of AISI 304 austenitic stainless steel," *Journal of Intelligent Manufacturing - J INTELL MANUF,* vol. 21, pp. 1-12, 06/01 2012.

[9] K. Salahshoor, M. Kordestani, and M. S. Khoshro, "Fault detection and diagnosis of an industrial steam turbine using fusion of SVM (support vector machine) and ANFIS (adaptive neuro-fuzzy inference system) classifiers," *Energy,* vol. 35, pp. 5472-5482, 2010/12/01/ 2010.

[10] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, "Machine Learning for Predictive Maintenance: A Multiple Classifier Approach," *IEEE Transactions on Industrial Informatics,* vol. 11, pp. 812-820, 2015.

[11] Z. Li, Z. Zhang, J. Shi, and D. Wu, "Prediction of surface roughness in extrusion-based additive manufacturing with machine learning," *Robotics and Computer-Integrated Manufacturing,* vol. 57, pp. 488-495, 2019/06/01/ 2019.

[12] M. Khanzadeh, P. Rao, R. Jafari-Marandi, B. K. Smith, M. A. Tschopp, and L. Bian, "Quantifying Geometric Accuracy With Unsupervised Machine Learning: Using Self-Organizing Map on Fused Filament Fabrication Additive Manufacturing Parts," *Journal of Manufacturing Science and Engineering,* vol. 140, 2017.

[13] L. Monostori, J. Hornyák, C. Egresits, and Z. J. Viharos, "Soft Computing and Hybrid AI Approaches to Intelligent Manufacturing," presented at the Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial In telligence and Expert Systems: Tasks and Methods in Applied Artificial Intelligence, 1998.

[14] D. Kim, P. Kang, S. Cho, H.-j. Lee, and S. Doh, "Machine learning-based novelty detection for faulty wafer detection in semiconductor manufacturing," *Expert Systems with Applications,* vol. 39, pp. 4075-4083, 2012/03/01/ 2012.

[15] T.-S. Li and C.-L. Huang, "Defect spatial pattern recognition using a hybrid SOM-SVM approach in semiconductor manufacturing," *Expert Syst. Appl.,* vol. 36, pp. 374-385, 01/31 2009.

[16] S. Doltsinis, P. Ferreira, and N. Lohse, "Reinforcement Learning for Production Ramp-Up: A Q-Batch Learning Approach," in *2012 11th International Conference on Machine Learning and Applications*, 2012, pp. 610-615.

[17] R. S. Sutton and A. G. Barto, *Reinforcement Learning, second edition: An Introduction*: MIT Press, 2018.

[18] T. Jacobs, F. Alesiani, and G. Ermis, "Reinforcement Learning for Route Optimization with Robustness Guarantees," in *IJCAI*, 2021.

[19] H. Wang, H. Shen, Q. Liu, K. Zheng, and J. Xu, "A Reinforcement Learning Based System for Minimizing Cloud Storage Service Cost," presented at the 49th International Conference on Parallel Processing - ICPP, Edmonton, AB, Canada, 2020.
[20] P. Ruvolo, I. R. Fasel, and J. R. Movellan, "Optimization on a Budget: A Reinforcement Learning Approach," in *NIPS*, 2008.
[21] L. Monostori, B. C. Csáji, and B. Kádár, "Adaptation and Learning in Distributed Production Control," *CIRP Annals,* vol. 53, pp. 349-352, 2004/01/01/ 2004.
[22] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez*, et al.*, "Mastering the game of Go without human knowledge," *Nature,* vol. 550, pp. 354-359, 2017/10/01 2017.
[23] J. Guevara, R. Patel, and J. Trivedi, *Optimization of Steam Injection for Heavy Oil Reservoirs Using Reinforcement Learning*, 2018.
[24] F. Hourfar, H. J. Bidgoly, B. Moshiri, K. Salahshoor, and A. Elkamel, "A reinforcement learning approach for waterflooding optimization in petroleum reservoirs," *Engineering Applications of Artificial Intelligence,* vol. 77, pp. 98-116, 2019/01/01/ 2019.
[25] J. He, M. Tang, C. Hu, S. Tanaka, K. Wang, X.-H. Wen*, et al.*, "Deep Reinforcement Learning for Generalizable Field Development Optimization," *SPE Journal,* vol. 27, pp. 226-245, 2022.
[26] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with EM-based Reinforcement Learning," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 3232-3237.
[27] P. Thomas, M. Branicky, A. van den Bogert, and K. Jagodnik, "Application of the Actor-Critic Architecture to Functional Electrical Stimulation Control of a Human Arm," *Proceedings of the ... Innovative Applications of Artificial Intelligence Conference / sponsored by the American Association for Artificial Intelligence. Innovative Applications of Artificial Intelligence Conference,* vol. 2009, pp. 165-172, 01/01 2009.
[28] J. Günther, P. M. Pilarski, G. Helfrich, H. Shen, and K. Diepold, "Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning," *Mechatronics,* vol. 34, pp. 1-11, 2016/03/01/ 2016.
[29] B. Waschneck, A. Reichstaller, L. Belzner, T. Altenmüller, T. Bauernhansl, A. Knapp*, et al.*, "Deep reinforcement learning for semiconductor production scheduling," in *2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, 2018, pp. 301-306.
[30] N. Stricker, A. Kuhnle, R. Sturm, and S. Friess, "Reinforcement learning for adaptive order dispatching in the semiconductor industry," *CIRP Annals,* vol. 67, pp. 511-514, 2018/01/01/ 2018.
[31] T. Altenmüller, T. Stüker, B. Waschneck, A. Kuhnle, and G. Lanza, "Reinforcement learning for an intelligent and autonomous production control of complex job-shops under time constraints," *Production Engineering,* vol. 14, pp. 319-328, 2020/06/01 2020.
[32] Y. H. Lee and S. Lee, "Deep reinforcement learning based scheduling within production plan in semiconductor fabrication," *Expert Systems with Applications,* vol. 191, p. 116222, 2022/04/01/ 2022.
[33] D. Shi, W. Fan, Y. Xiao, T. Lin, and C. Xing, "Intelligent scheduling of discrete automated production line via deep reinforcement learning," *International Journal of Production Research,* vol. 58, pp. 3362-3380, 2020/06/02 2020.
[34] S. Luo, "Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning," *Applied Soft Computing,* vol. 91, p. 106208, 2020/06/01/ 2020.
[35] I. Park, J. Huh, J. Kim, and J. Park, "A Reinforcement Learning Approach to Robust Scheduling of Semiconductor Manufacturing Facilities," *IEEE Transactions on Automation Science and Engineering,* vol. 17, pp. 1420-1431, 2020.
[36] D. J. Pradeepa and M. M. Noel, "A Finite Horizon Markov Decision Process Based Reinforcement Learning Control of a Rapid Thermal Processing system," *Journal of Process Control,* vol. 68, pp. 218-225, 2018.
[37] Y. Li, J. Du, and W. Jiang, "Reinforcement Learning for Process Control with Application in Semiconductor Manufacturing," *arXiv:2110.11572,* 2021.
[38] M. Khakifirooz, M. Fathi, and C.-F. Chien, "Partially Observable Markov Decision Process for Monitoring Multilayer Wafer Fabrication," *IEEE Trans. Autom. Sci. Eng.,* vol. 18, pp. 1742-1753, 2021.
[39] N. Khader and S. W. Yoon, "Adaptive optimal control of stencil printing process using reinforcement learning," *Robotics and Computer-Integrated Manufacturing,* vol. 71, p. 102132, 2021.
[40] M. Wiering and M. Otterlo, *Reinforcement Learning: State-Of-The-Art* vol. 12. Berlin: Springer, 2012.
[41] Y.-C. Wang and J. M. Usher, "Application of reinforcement learning for agent-based production scheduling," *Eng. Appl. Artif. Intell.,* vol. 18, pp. 73–82, 2005.
[42] C. C. H. Chung-Yuan Chang, Tejender Rawat, Shih-Wei Chen, Albert Lin, "Human machine competition in intelligent laser manufacturing in semiconductor processes," in *SPIE Optics + Photonics*, San Diego, California, United States, 2022.
[43] A. L. Robinson, "Laser Annealing: Processing Semiconductors Without a Furnace," *Science,* vol. 201, pp. 333-335, 1978.
[44] A. a. S. Kuhnle, Michael and Fricke, Kai. (2017). *Tensorforce: a TensorFlow library for applied reinforcement learning*. Available: https://github.com/tensorforce/tensorforce
[45] Y. Berthelot. (2020). *AI learns to fly | Airplane simulation and Reinforcement Learning*. Available: https://github.com/YannBerthelot/PlaneModel
[46] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro*, et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," Google Res., USA, 2015.
[47] G. V. Rossum and F. L. Drake, *Python 3 Reference Manual*: CreateSpace, 2009.
[48] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau*, et al.*, "Array programming with NumPy," *Nature,* vol. 585, pp. 357-362, 2020/09/01 2020.
[49] E. Jones, T. Oliphant, and P. Peterson, "SciPy: Open source scientific tools for Python," 2001.
[50] W. McKinney, "Data Structures for Statistical Computing in Python," 2010.
[51] L. Yiu-Wing and W. Yuping, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Transactions on Evolutionary Computation,* vol. 5, pp. 41-53, 2001.
[52] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *Computers & Operations Research,* vol. 134, p. 105400, 2021/10/01/ 2021.
[53] *Deep Reinforcement Learning, Fundamentals, Research and Applications*: Springer, 2020.
[54] B. El-Kareh, *Fundamentals of Semiconductor Processing Technology*: Springer New York, NY, 1994.
[55] J. Choma, "The computation of semiconductor sheet resistance," *IEEE Transactions on Electron Devices,* vol. 32, pp. 845-847, 1985.
[56] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, "Combining model-based and model-free updates for trajectory-centric reinforcement learning," presented at the Proceedings of the 34th International Conference on Machine Learning - Volume 70, Sydney, NSW, Australia, 2017.
[57] A. Nagabandi, G. Kahn, R. Fearing, and S. Levine, "Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning," presented at the Deep Reinforcement Learning Symposium, NIPS 2017, Long Beach, CA, USA., 2018.
[58] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal Value Function Approximators," presented at the Proceedings of the 32nd International Conference on Machine Learning, Proceedings of Machine Learning Research, 2015.
[59] Y. W. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell*, et al.*, "Distral: robust multitask reinforcement learning," presented at the Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, California, USA, 2017.
[60] L. Mai, N.-N. Dao, and M. Park, "Real-Time Task Assignment Approach Leveraging Reinforcement Learning with Evolution Strategies for Long-Term Latency Minimization in Fog Computing," *Sensors,* vol. 18, p. 2830, 2018.

[61] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *arXiv preprint arXiv:1712.06567,* 2017.