Deep Reinforcement Learning-based Approach for Efficient and Reliable Droplet Routing on MEDA Biochips*

Mahmoud Elfar, Yi-Chen Chang, Harrison Hao-Yu Ku, Tung-Che Liang,
Krishnendu Chakrabarty, and Miroslav Pajic

***Abstract*—The micro-electrode-dot-array (MEDA) architecture provides precise droplet control and real-time sensing in digital microfluidic biochips. Previous work has shown that trapped charge under microelectrodes leads to droplets being stuck and failures in fluidic operations. A recent approach utilizes real-time sensing of microelectrode health status, and attempts to avoid degraded electrodes during droplet routing. However, the problem with this solution is that the computational complexity is unacceptable for MEDA biochips of realistic size. Consequently, in this work, we introduce a deep reinforcement learning (DRL)-based approach to bypass degraded electrodes and enhance the reliability of routing. The DRL model utilizes the information of health sensing in real-time to proactively reduce the likelihood of charge trapping and avoid using degraded microelectrodes. Simulation results show that our approach provides effective routing strategies for COVID-19 testing protocols. We also validate our DRL-based approach using fabricated prototype biochips. Experimental results show that the developed DRL model completed the routing tasks using a fewer number of clock cycles and shorter total execution time, compared with a baseline routing method. Moreover, our DRL-based approach provides reliable routing strategies even in the presence of degraded electrodes. Our experimental results show that the proposed DRL-based routing is robust to occurrences of electrode faults, as well as increases the lifetime and usability of microfluidic biochips compared to existing strategies.**

I. Introduction

As microfluidic technology advances, digital microfluidic biochips (DMFBs) are being utilized to automate laboratory procedures associated with bioanalytical assays. DMFBs revolutionize traditional experimental processes by providing precise control of nanoliter-sized droplets. Various bio-experiments have been performed using DMFBs, such as molecular detection, diagnostic tests for newborns, portable detection for COVID-19, and aerosol detection [1], [2], [3], [4], [5], [6].

In recent years, an improved DMFB structure called micro-electrode-dot-array (MEDA) has been proposed. MEDA biochips adopt the concept of sea-of-micro-electrodes and consist of a large number of microelectrodes (MCs), where each MC contains an individual circuit for real-time control and sensing [7], [8]. In contrast to conventional DMFBs, MEDA biochips are implemented using a $0.35\,\mu\mathrm{m}$ standard CMOS process [9], [10]. The size of an MC is around 40 times smaller than the size of an electrode in a conventional DMFB. Hence, the MCs can be flexibly grouped to form different types of microfluidic modules during bioprotocol execution on MEDA biochips.

Both conventional DMFBs and MEDA biochips adopt the principle of electrowetting-on-dielectric (EWOD) to manipulate droplets. A high voltage is repeatedly applied to the electrodes to generate dragging forces on the droplets. However, as the electrodes are frequently charged and discharged, charge trapping might occur on the electrodes and result in electrode degradation. The degraded electrodes generate abnormal EWOD forces, which cause unexpected droplet movements (or lack thereof) and lead to failure of microfluidic operations. Therefore, reliability is a major concern for microfluidic biochips. While solutions have been proposed in the literature to mitigate the problem of electrode degradation, most of these methods focus on recovery after erroneous behaviors occur. These techniques include the design of fluidic checkpoints, droplet-aliquot operations, synthesis of recovery protocols using probabilistic-timed-automata, as well as selective sensing [11], [12], [13], [14], [15], [16]. On the other hand, preventive approaches that proactively predict and prevent failures during the droplet routing stage have not received much attention.

Recently, a reinforcement learning (RL)-based droplet routing model has been developed to address the reliability problems in DMFBs [17]. However, the main limitation of this method is that the RL developed model can only learn from the occurrence of failures. In other words, the model adjusts its policy only after an error has occurred. This limitation prevents the model from being applicable to time-sensitive applications such as flash chemistry [18].

A new hardware design, which enables real-time sensing of health status for each MC on MEDA biochips, was first introduced in [19]. A stochastic game-based formal synthesizer was proposed to generate adaptive routing strategies based on the electrode health information derived from MEDA biochips in real-time [20]. However, the formal synthesizer suffers from the limitation of scalability in terms of biochip size. For a MEDA biochip of size 20×20, around three seconds are needed for the formal synthesizer to complete a synthesis task for one step, and a bioassay consists of at least hundreds of

* This research was supported in part by the National Science Foundation under grant No. ECCS-1914796.

Mahmoud Elfar, Yi-Chen Chang, Harrison Hao-Yu Ku, Krishnendu Chakrabarty, and Miroslav Pajic are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA (e-mail: mahmoud.elfar@duke.edu; yichen.chang@duke.edu; harrison.hy.ku@duke.edu; tung.che.liang@duke.edu; krish@duke.edu; miroslav.pajic@duke.edu).

such steps. Therefore, the formal synthesizer is infeasible for practical applications. For instance, a commercial microfluidic biochip platform called aQdrop includes 41 thousand electrodes [21], which is around 100 times larger than the size of the biochips used in [20]. Thus, a formal synthesis method that explores all the state-space cannot be deployed for state-of-the-art biochip platforms.

Consequently, in this work, we introduce a DRL-based droplet routing approach that incorporates real-time health information to provide routing strategies that proactively avoid the use of degraded MCs. The routing framework is deployed with a deep neural network (DNN) agent, which is first trained using offline DRL with simulated environments. Then, online DRL training is applied to the agent to adjust the policy under different biochip environments. In contrast to the formal synthesizer from [20], which enumerates all the possible state spaces, the DRL model efficiently stores routing solutions using the DNN agent. The agent provides routing strategies with negligibly short computing time. Thus, the proposed DRL model can be employed under realistic scenarios for different applications.

A preliminary version of this DRL-based routing approach appeared in [22]. In this paper, we present more details on the DRL model and the training methods for CNN agents. We have also validated the proposed DRL routing model using fabricated prototype biochips. Specifically, the main contributions of this paper are as follows:

- We propose a DRL-based approach (and the corresponding DRL model) for droplet routing on MEDA biochips, providing reliable routing strategies based on real-time health information.
- We introduce a stochastic model to constitute the virtual training environments for the DRL model. To enhance training efficiency, we adopt action space parameterization and adaptive droplet step movement.
- We design a DNN agent for the DRL model that ensures the scalability of the DRL framework. We deploy both traditional and transfer learning techniques for agent training, and evaluate their performance.
- We validate the estimation of degradation parameters using PCB-based biochips.
- We compare the performance of the DRL model with the formal synthesizer by running bioassays for COVID-19 testing.
- We design and execute experiments where we evaluate the trained DNNs on PCB-based biochips, and compare the proposed solution to baseline routing policies.

The remainder of this paper is organized as follows. Section II provides background on MEDA biochips, adaptive routing, reinforcement learning, and the problem formulation for the adaptive droplet routing problem. Section III introduces the proposed DRL model for adaptive droplet routing. Section IV describes the configurations used for training, the DNN architecture, the training algorithm, and two approaches for training multiple DNNs. Section V evaluates the degradation parameters and compares the performance of the proposed DRL framework to existing routing methods. Section VI describes the experiment design and presents results for evaluating the trained agents using PCB-based biochips. Finally, Section VII concludes the paper.

II. Background and Motivation

A. MEDA Biochips

MEDA biochips apply the EWOD [23] mechanism to manipulate individual droplets with droplet volumes in a nano-liter scale. Typically, one biochip may contain up to thousands of MCs. Every MC module is composed of three components: a microelectrode, a controlling unit, and a capacitance-sensing circuit for real-time detection of droplet location. Depending on the bioassay requirements, the controller can dynamically reconfigure the grouping of microelectrodes, and thus modules such as mixers and splitters can be formed. When a sensing operation is conducted, a charging and discharging process is applied to all the MCs to measure the capacitance difference and thus the locations of droplets can be determined. A scan-chain architecture is adopted to connect all the MCs into a daisychain, whereby the control signals and the sensing data are shifted in and out as a sequences of bits.

B. Adaptive Droplet Routing

As biochip platforms such as Illumina, Genmark, and Baebies [24], [25], [26], are commercialized by companies, the reliability of these devices has emerged as a major concern. Manufacturing defects can be detected using the method proposed in [27]. However, degradation due to charge trapping in the dielectric layer occurs when electrodes are used over time. The degraded electrodes lead to the failure of droplet movement when they are involved in the transportation path. To overcome reliability problem, a wear-leveling method has been proposed to avoid electrode overuse by uniformly distributing the electrode usage during the mapping of the fluidic operations to the biochips [28]. However, for droplet routing, only a few methods consider the dynamic changes of the health condition of the biochips.

The first RL-based routing model that considers electrode degradation on DMFBs was introduced in [17]. However, this method is limited to conventional DMFBs, in which the unique features for MEDA biochips are not incorporated. For instance, the model only allows droplets of a single size while MEDA biochips support droplets of various sizes. Another limitation of [17] is that this model needs to learn from occurrence of failures, which might lead to erroneous outcomes of bioassay execution.

Recently, a new MC design for MEDA biochips was proposed in [20]. This design enables real-time sensing of the health information of individual MCs. A formal synthesizer has been developed to utilize health sensing to provide routing strategies that maximize the likelihood of successful bioassay execution. An analytical model that defines the probabilistic behaviors between the degradation level of MCs and the corresponding probability of a successful actuation has also been developed. Based on this analytical model, the synthesizer provides an optimal routing strategy using a stochastic game-based formulation. However, the computing time of this

method grows significantly as the size of biochip increases. Hence, the large runtime is unacceptable for realistic applications.

C. Reinforcement Learning

Reinforced learning (RL) is a class of machine learning in which the models learn the optimal strategy in an complicated environment. RL algorithms have been shown to be efficient for applications across various domains, such as robotic manipulation, playing GO, video games and medical devices [29], [30], [31], [32]. An RL problem can be formulated using Markov decision processes (MDPs) using a tuple $(\mathcal{S}, \mathcal{A}, T, R)$, where $\mathcal{S}$ is the set of states; A is the set of actions; $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition function; and $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. The aim of an RL agent is to find a policy that maximizes the total reward received from interactions with an uncertain environment at discrete time steps. In modern RL applications, DNN agents are deployed to obtain a near-optimal policy under large state and action spaces.

D. Problem Formulation

In this work, we address the problem of designing droplet routing policies for MEDA biochips. We assume that a bioassay scheduler breaks down microfluidic operations into a series of single-droplet routing jobs. Each routing job is characterized by droplet shape and size, initial and final droplet locations, and the biochip area within which routing is allowed.

Then, the objective is to design routing policies for droplets to successfully execute a given routing job. At each control cycle, a routing policy shall provide the actuation pattern to be applied. To this end, the routing policy is expected to utilize both the real-time droplet sensing feedback, as well as the microelectrode health feedback provided by the onboard sensors.

We also consider the issue of designing such routing policies for multiple biochip sizes and fault injection levels. From a run-time perspective, a routing policy need to be available within few seconds of receiving the routing job. Furthermore, at each control cycle, the time between receiving sensor measurements and generating the control pattern should not exceed 200 ms. Those run-time requirements are motivated by typical applications where the control cycle period is 1 sec. In addition, time-critical microfluidic applications — such as flash chemistry [18] — demand timely response on the scale of fractions of few seconds.

III. DRL Approach for MEDA

In this section, we present the DRL-based framework for designing routing policies for MEDA biochips. We first introduce the stochastic model that constitutes the virtual training environment. We then discuss the elements comprising the training environment; specifically, the action space, the observation space, and the reward function.

For notation, $\mathbb{Z}$, $\mathbb{N}_0$ and $\mathbb{R}$ denote the set of integer, natural, and real numbers, respectively. We use $\mathbb{1} : \{\top, \bot\} \to \{0, 1\}$ to denote the indicator function over the set of Booleans. We also utilize $\mathcal{U}\{i, j\}$ and $\mathcal{U}(a, b)$ to denote the discrete (i.e., over integers) and the continuous (i.e., over reals) uniform distributions over $[\![i, j]\!]$ and $[a, b]$, respectively. Finally, we use bold capital letters for matrices, and italic capital letters for their elements.

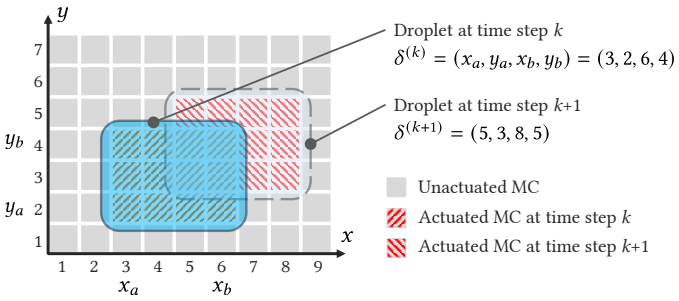


Fig. 1: Example for droplet coordinates at time steps k and $k + 1$.

A. MEDA Training Environment

Consider a MEDA biochip of size $\mathsf{W} \times \mathsf{H}$, denoting the number of MCs in each row and column, respectively. Following [19], we model a droplet as a quadruple $\delta = (x_a, y_a, x_b, y_b) \in \Delta$, where $\Delta \subset \mathbb{N}_0^4$ is the set of all possible droplets. A routing task is characterized by the initial (start) and target (goal) droplet locations, denoted by δ_s and δ_g, respectively. We use $\delta^{(k)}$, $k \in \mathbb{N}_0$, to denote the droplet location at the kth control step. Fig. 1 shows an example of the droplet location.

Let (i, j) be the coordinates of a given MC; $D_{ij} \in [0, 1]$ be its degradation level, where 1 indicates a fully healthy MC and 0 a fully degraded; and $n_{ij} \in \mathbb{N}_0$ be the total number of control steps at which the MC was actuated. The degradation level can be estimated as $D_{ij}^{(n)} = \tau_{ij}^{n_{ij}/c_{ij}} \in [0, 1]$, where $c_{ij} \in \mathbb{R}_{>0}$ and $\tau_{ij} \in [0, 1)$ are parameters controlling the degradation rate. Those parameters are generally unknown, although their range can be experimentally estimated [33]. The degradation level of an MC can be measured through the health measurement unit [19]. Given a health measurement unit with b-number of bits, the MC measured health is captured by

$$H_{ij}^{(n)} = \lfloor 2^b \cdot D_{ij}^{(n)} \rfloor = \lfloor 2^b \cdot \tau^{n_{ij}/c_{ij}} \rfloor. \tag{1}$$

At each control cycle, MEDA biochips support single- and double-step droplet movements in both the cardinal and ordinal directions. The movement is achieved by applying an actuation pattern that corresponds to the movement direction, and the probability that this movement is successful largely depends on the health level of the group of microelectrodes — referred to as the *frontier set* — primarily responsible for generating the EWOD force for the action to be performed. We employ the probabilistic transitions modeling from [19]. Each action, along with the current droplet location, determines the group of microelectrodes to be actuated. We use $\mathbf{U}^{(k)} \in \{0, 1\}^{\mathsf{W} \times \mathsf{H}}$ to denote the actuation pattern matrix (pattern, for short) applied at time step k, where $U_{ij} = 1$ indicates that the microelectrode $\mathrm{MC}_{(i,j)}$ is actuated.

Algorithm 1: Procedure for Computing Number of Steps

Input: Droplet δ: (x_a, y_a, x_b, y_b); goal δ_g: $(x_{ag}, y_{ag}, x_{bg}, y_{bg})$; action $a \in \mathcal{A}$
Output: Signed distance (λ_x, λ_y)

$(\lambda_x, \lambda_y) \leftarrow (0, 0)$
$(\Delta x, \Delta y) \leftarrow (x_{ag} - x_a, y_{ag} - y_a)$
$(\Lambda_x, \Lambda_y) \leftarrow (\lfloor (x_b - x_a + 1)/2 \rfloor, \lfloor (y_b - y_a + 1)/2 \rfloor)$
if $a \in \{a_{\mathsf{N}}, a_{\mathsf{NE}}, a_{\mathsf{NW}}\}$ **then**
  $\lambda_y \leftarrow \Lambda_y - (\Lambda_y - \Delta y) \cdot \mathbb{1}_{\{0 < \Delta y < \Lambda_y\}}$
if $a \in \{a_{\mathsf{S}}, a_{\mathsf{SE}}, a_{\mathsf{SW}}\}$ **then**
  $\lambda_y \leftarrow -\Lambda_y + (\Lambda_y + \Delta y) \cdot \mathbb{1}_{\{-\Lambda_y < \Delta y < 0\}}$
if $a \in \{a_{\mathsf{E}}, a_{\mathsf{NE}}, a_{\mathsf{SE}}\}$ **then**
  $\lambda_x \leftarrow \Lambda_x - (\Lambda_x - \Delta x) \cdot \mathbb{1}_{\{0 < \Delta x < \Lambda_x\}}$
if $a \in \{a_{\mathsf{W}}, a_{\mathsf{NW}}, a_{\mathsf{SW}}\}$ **then**
  $\lambda_x \leftarrow -\Lambda_x + (\Lambda_x + \Delta x) \cdot \mathbb{1}_{\{-\Lambda_x < \Delta x < 0\}}$
return (λ_x, λ_y)

B. Parameterized Action Space

In a traditional action space, an action captures both the direction and magnitude of the movement. This results in a large action space cardinality, rendering the model unsuitable for training. For instance, an action space that supports double-step movements is comprised of at least 16 actions. Hence, we propose a parameterized action space where actions capture only the movement direction, while the number of steps is defined based on the droplet size, shape, and its location relative to the goal. The motivation behind the parameterization of the action space is twofold. First, it reduces the dimensionality of the model by reducing the action space size. Second, it unifies the action set across different droplet shapes and sizes, enabling the usage of one trained agent for the entire range of droplet sizes. Consequently, a parameterized action space is an efficient representation that allows for moving a droplet beyond two steps at a time.

We define the parameterized action space as the set $\mathcal{A} = \{a_{\mathsf{N}}, a_{\mathsf{S}}, a_{\mathsf{E}}, a_{\mathsf{W}}, a_{\mathsf{NE}}, a_{\mathsf{NW}}, a_{\mathsf{SE}}, a_{\mathsf{SW}}\}$, where N, S, E and W stands for north, south, east and west, respectively. Let $(\lambda_x, \lambda_y) \in \mathbb{Z}$ be the signed distance (distance, for short) associated with the adaptive action $a \in \mathcal{A}$. Algorithm 1 presents the procedure for computing (λ_x, λ_y) given the current droplet location δ, goal location δ_g, and action a. Basically, the procedure computes the maximum movement distance based on the droplet size and the movement direction, while avoiding overshooting the goal location. The computed distance (λ_x, λ_y) is then used to transform the action a into the corresponding actuation pattern $\mathbf{U}$. Note that $|\mathcal{A}| = 8$ at all states, reducing the complexity of the model and, subsequently, the time required for training.

Example 1. *The droplet shown in Fig. 1 is of size* 4×3. *Since the maximum reliable distance for the droplet to travel is* $(\lambda_x, \lambda_y) = (\lfloor w/2 \rfloor, \lfloor h/2 \rfloor) = (2, 1)$, *the adaptive action* a_{NE} *attempts to move the droplet one and two steps in the east and north directions, respectively, during the current control cycle. If the goal location is* $\delta_g = (4, 3, 7, 5)$, *then the distance is capped at* $(\lambda_x, \lambda_y) = (1, 1)$ *to prevent the droplet from overshooting.*

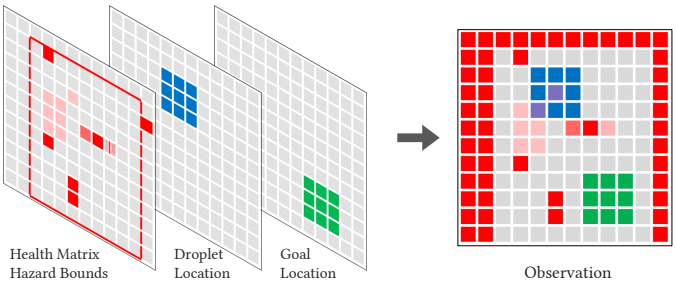


Fig. 2: Channels comprising the observation space.

C. Observation Space

At each control step k, the DRL agent can observe the current sensor matrix $\mathbf{Y} \in \{0, 1\}^{\mathsf{W} \times \mathsf{H}}$. For a droplet $\delta = (x_a, y_a, x_b, y_b)$, $Y_{ij} = 1$ for all $(i, j) \in [\![x_a, x_b]\!] \times [\![y_a, y_b]\!]$, and $Y_{ij} = 0$ otherwise. In addition, the DRL agent can read the health matrix $\mathbf{H}^{(k)} \in \{0, 1, \ldots, 2^b - 1\}^{\mathsf{W} \times \mathsf{H}}$. From a DRL perspective, an observation shall also incorporate the current droplet location δ, goal location δ_g, and the hazard bounds δ_h.

To preserve the spatial relationships among the observed data, we utilize a 3D image-based observation space. Specifically, we define an observation as a 3D matrix $o \in [0, 1]^{\mathsf{H} \times \mathsf{W} \times 3)}$. As shown in Fig. 2, the first layer captures both the health matrix and hazard bounds, and is defined as

$$o(i, j, 1) = \begin{cases} H(i, j, 1)/2^b & i \in [x_{ah}, x_{bh}], j \in [y_{ah}, y_{bh}], \\ 0 & \text{otherwise}, \end{cases}$$

where the hazard bounds are indirectly captured by masking the health matrix values outside those bounds. The second layer is defined as

$$o(i, j, 2) = \begin{cases} 1 & i \in [x_a, x_b], j \in [y_a, y_b], \\ 0 & \text{otherwise}, \end{cases}$$

to capture the droplet location. Similarly, the third layer is defined as

$$o(i, j, 3) = \begin{cases} 1 & i \in [x_{ag}, x_{bg}], j \in [y_{ag}, y_{bg}], \\ 0 & \text{otherwise}, \end{cases}$$

to capture the goal location. Note that the elements of the first layer are scaled so that the observation elements are both within the range $[0, 1]$ and independent of the actual number of bits b used for health measurements.

D. Reward Function

The primary goal in adaptive droplet routing is to ensure that the droplet can reach the target location. Performance metrics in this case include the time and distance traveled by the droplet. Since excessive actuations of individual microelectrodes can lead to their premature failure, the number of actuations per microelectrode has to be incorporated in the routing process.

Let $a^{(k)}$ be the action taken at step k from state $s^{(k)}$, resulting in a new state $s^{(k+1)}$. Thus, the reward is defined as

$$r(k) = \alpha_{\text{dis}} r_{\text{dis}}^{(k)} + \alpha_{\text{ter}} r_{\text{ter}}^{(k)} + \alpha_{\text{act}} r_{\text{act}}^{(k)}$$

where r_{dis}, r_{deg} and r_{ter} are the distance, terminal and action rewards, respectively, and $\alpha_i \in \mathbb{R}$ are the respective

hyperparameters. To incentivize progression towards the target location, r_{dis} is defined as

$$r_{\text{dis}}^{(k)} = D\left(\delta^{(k)}, \delta_g\right) - D\left(\delta^{(t+1)}, \delta_g\right),$$

where $D\left(\delta^{(k)}, \delta_g\right)$ denotes the Manhattan distance between two droplet locations. The terminal reward r_{ter} aids in faster convergence by associating reaching the target location with an additional reward, defined as $r_{\text{ter}}^{(k)} = \mathbb{1}\{\delta = \delta_g\}$. Finally, the action reward r_{act} penalizes selecting an invalid action, i.e., an action that causes the droplet to exit the routing job area. The selection of the hyperparameters α_i is discussed in Section IV.

Note that a maximum number of cycles per routing job is imposed during training to allow for diverse sampling. While the agent is rewarded for reaching the target location, it is not penalized if the routing job fails due to reaching the maximum number of cycles allowed. The reason is that the number of cycles available for routing is not part of the observation space. That is, states that only differ by the number of cycles remaining have identical observations, leading to state aliasing [34].

IV. DRL Agent Design and Training

This section summarizes our approach for design of the DRL agents, to be employed for routing, including the employed architecture and training procedure.

A. DNN Architecture and Training Configurations

We first discuss the employed DNN architecture as well as configuration parameters that affect the training convergence speed – i.e., MEDA biochip size, droplet size, the initial and target droplet locations, the initial microelectrode degradation levels, and the degradation parameters.

DNN Architecture: We employ a convolutional neural network (CNN) to learn droplet routing policies due to its potential in preserving important features of the observation space. As illustrated in Fig. 2, the input to the CNN is a matrix of size $(\mathsf{H}, \mathsf{W}, 3)$. The three channels represent the microelectrode health levels and routing zone, the goal location, the current droplet location. The agent's goal is to learn a policy that maximized the expected cumulative reward. For notation, we use H and H^* to denote untrained and trained CNN agents, respectively.

Biochip and Droplet Sizes: For training, we considered biochips of sizes between 30×30 and 180×180. We trained the agent for the most common droplet sizes, with droplet width and height $w, h \in \{2, 3, 4, 5, 6\}$, where $w/h \in [0.8, 1.25]$. We assume that the droplet size is preserved throughout a single routing job. Hence, there are two approaches to droplet size selection during training. In the first, multiple agents are utilized, where each agent is trained for a specific droplet size. In the second, the same agent is trained against the range of droplet sizes. Note that a DNN can be feature-invariant by training against the range of values for such feature. Moreover, the exact size of droplets during execution might slightly vary outside those specific values. Consequently, we opt for training a single agent in this framework (i.e., the second approach).

TABLE I: CNN layers and their configurations.

Layer	Type	Activation	Size	Stride	Padding
L1	Convolution	ReLU	64	3	1
L2	Convolution	ReLU	128	3	1
L3	Convolution	ReLU	128	3	1
L4	Fully-connected	ReLU	256	3	1
L5	Output	ReLU	8	–	–

Degradation Parameters: From (1), degradation parameters of microelectrodes affect their degradation rate, although they are not directly observable to the agent. For training, we randomly sample the degradation parameters as $c_{ij} \sim \mathcal{U}(c_{\min}, c_{\max})$ and $\tau_{ij} \sim \mathcal{U}(\tau_{\min}, \tau_{\max})$, where the distributions are experimentally obtained as described in Section V. On the other hand, the number actuations n_{ij} is updated based on the actuation patterns applied by the agent at each step.

Initial and Target Locations: In MEDA biochips, a droplet is either the result of a preceding microfluidic operation or dispensed by an on-chip dispenser. In the former case, the droplet location can be anywhere on the biochip; in the latter, the initial location δ_s is one of multiple, predefined dispenser coordinates. Similarly, the target location δ_g can be either where a microfluidic module is (e.g., a mixer or a splitter), or a predefined exit through one of the biochip reservoirs.

For benchmark bioassays, the percentage of routing jobs involving initial (e.g., dispensing operations) or target (e.g., discarding operations) droplets adjacent to one of the biochip edges is between 20% and 40% [35]. Thus, during training both the initial and goal locations are sampled from a stratified distribution. Specifically, we randomly sample δ_s and δ_g at the start of each training episode such that $x_{as}, x_{ag} \sim \mathcal{U}\{2, \mathsf{W}\text{–}w\text{–}1\}$, and $y_{as}, y_{ag} \sim \mathcal{U}\{2, \mathsf{H}\text{–}h\text{–}1\}$.

B. Agent Training

For training, we use the proximal policy optimization (PPO) algorithm [36], [37] with the actor-critic architecture. Unlike policy gradient methods for reinforcement learning where policy gradients are updated after reading each data sample, PPO utilizes a surrogate objective to stabilize the training process via multiple workers. Algorithm 2 summarizes the procedure for training the agent to learn droplet routing policies. Each training episode starts with the initial state $(\delta_g, \mathbf{H})$, sampled according to the distributions described earlier. After each step, the number of actuations n_{ij} is updated for all MCs using the actuation pattern matrix $\mathbf{U}$.

An episode terminates whenever one of two conditions is met, namely, reaching a target location (i.e., $\delta = \delta_g$), or reaching a predefined threshold for the number of cycles, denoted by $k_{\max}$. Imposing the second condition aids in diversifying the routing jobs used in training. We use $k_{\max} = \alpha(\mathsf{W}_h + \mathsf{H}_h)$, where $\alpha \in [1, 2]$ is a hyperparameter, and W_h and H_h are the width and height of the hazard zone, respectively.

An episode is terminated when either the target location is reached, i.e., $\delta = \delta_g$, or the maximum number of steps allowed has passed, i.e., $k = 2(\mathsf{W}+\mathsf{H})$. Using the accumulated rewards, the gradients for each encountered (s, a) are updated.

Algorithm 2: Procedure for Learning Routing Policies

Input: MEDA biochip size
for *epoch* **do**
  $resample \leftarrow \top$
  for $iter = 1, 2, \ldots, N_{iter}$ **do**
    for $actor = 1, 2, \ldots, N_{actor}$, *running in parallel,* **do**
      if $resample = \top$ **then**
        Sample δ_s, δ_g, (τ_{ij}), (c_{ij}), and **N**
        $resample \leftarrow \bot$
      Run current policy π and obtain rewards and new state
      if $(k \geq k_{\max}) \vee (\delta = \delta_g)$ **then** $resample \leftarrow \top$
    if $(iter \cdot N_{actor}) \bmod minibatchsize = 0$ **then**
      Optimize PPO2 loss function, update current policy π

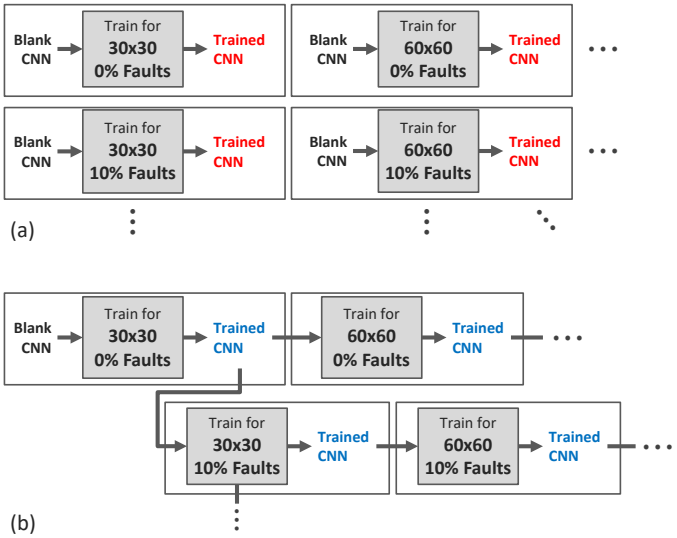


Fig. 3: Dataflow diagram for (a) traditional learning and (b) transfer learning.

To avoid catastrophic unlearning, we adopt a dynamic learning rate scheduler for training. Specifically, the training process starts with a base learning rate η_0. At the end of the i-th epoch, the learning rate is discounted with factor β_η only if the agent performance is above a certain threshold, i.e.,

$$\eta_{i+1} = \begin{cases} \max\left(\beta_\eta * \eta_i, \eta_{\min}\right) & \text{success rate} > 0.99, \\ \eta_i & \text{otherwise.} \end{cases}$$

Through hyperparameter optimization, we chose $\eta_0 = 3.5 \times 10^{-4}$, $\eta_{\min} = 1.0 \times 10^{-6}$, and $\beta_\eta = 0.7$.

C. Training Multiple CNNs

In this work, we explore two approaches for training multiple CNN agents as shown in Fig. 3. In the first approach, namely traditional learning, we train a CNN for each droplet size and fault injection level, where each training process starts with a randomly initialized CNN. Since there is no dependency between the various training processes, CNNs can employ observations of various sizes independently of each other.

In the second approach, namely transfer learning, we utilize pre-trained CNNs to accelerate the training process for untrained agents. Specifically, we first train a randomly initialized CNN $H_{(30,0)}$ on biochips of size 30×30 with no fault injection, resulting in $H^*_{(30,0\%)}$. Next, we use the pre-trained agent to initialize the training of the CNNs used for the next biochip size and fault injection level. For example, we use $H_{(60,0\%)} = H^*_{(30,0\%)}$ and $H_{(30,0.1)} = H^*_{(30,0\%)}$ to obtain the networks $H^*_{(60,0\%)}$ and $H^*_{(30,0.1)}$, respectively. The process is then repeated using the new CNNs as illustrated in Fig. 3.

In order to make the transfer learning process feasible, the input layer size is unified across all CNNs. This is achieved by scaling the observation matrix from the original size, i.e., $(\mathsf{W}, \mathsf{H}, 3)$, to a unified observation size $(30, 30, 3)$. The scaling is performed using an algorithm provided by OpenCV library [38] that resamples the original observation using pixel area relation.

Fig. 4 compares the training performance of the CNNs for various biochip sizes trained via traditional and transfer learning. For all biochip sizes, the transfer learning-based CNNs are able to learn effective policies within the first training epoch, exhibiting the same performance that the CNNs trained via random initialization were able to achieve after 15 to 40 epochs. This gain in training performance comes at the cost of computation required to resize the observations. Nevertheless, this cost is negligible when compared to the computational power and time required to train the CNNs for more epochs using observations of larger sizes. Another consideration is that training multiple CNNs using transfer learning cannot be fully done in parallel. Since the initial training is done offline, training one CNN at a time is considered acceptable.

V. Model and Learning Evaluation

A. Measurement and Modeling of Degradation Parameters

The first series of experiments aims to establish the degradation model and to evaluate the coefficients in (1). We monitored the processes of electrode degradation in PCB-based DMFBs, which utilize the same EWOD principle as MEDA biochips to manipulate droplets. Electrodes of three sizes are included on our biochips: $2{\times}2\,\mathrm{mm}^2$, $3{\times}3\,\mathrm{mm}^2$, and $4{\times}4\,\mathrm{mm}^2$ (see Fig. 5). Four reservoir modules on two sides of the DMFB are used to dispense different reagent droplets. The actuation of each electrode can be controlled individually using a high-voltage relay on the control board. Each high-voltage relay is controlled using a single configuration bit, and these configuration bits are stored in the shift register ICs.

Fig. 5 shows the overall hardware design of the DMFB and the controller. Identical actuation sequences are executed on two DMFBs simultaneously to accelerate the experimental process.

A series of actuation sequences are designed to simulate repeated bioassay executions on the biochips. The electrodes are activated and deactivated under a high frequency. The charging time is monitored via an oscilloscope after each execution. The charging path can be simplified as an RC circuit since the electrode and the top plate form a capacitor, and a resistor is connected in series between the electrodes. The effective capacitance of an electrode at time t can then be derived using $V_C(t) = V_{pp}\left(1 - e^{-t/RC}\right)$, where $V_C(t)$ is the electrode capacitance at time t. Subsequently, the EWOD force F can be obtained from [39], [40] as

$$F = \frac{C(V_C - V_T)^2}{2} \frac{dA(x)}{dx},$$

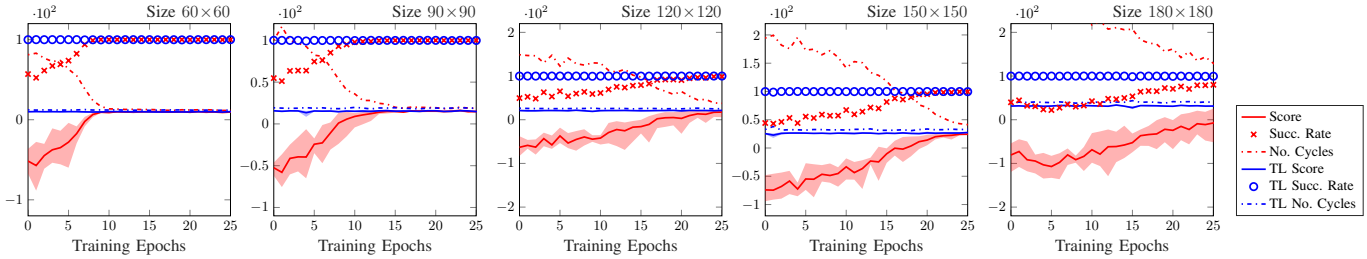


Fig. 4: Performance results for training CNNs via random initialization (red) and transfer learning (blue).

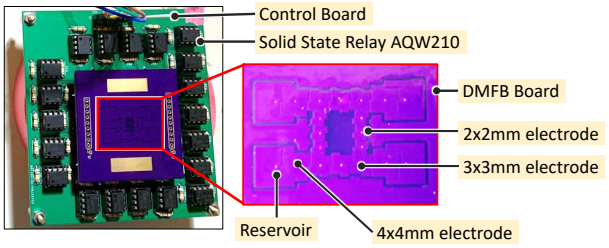


Fig. 5: The experimental setup.

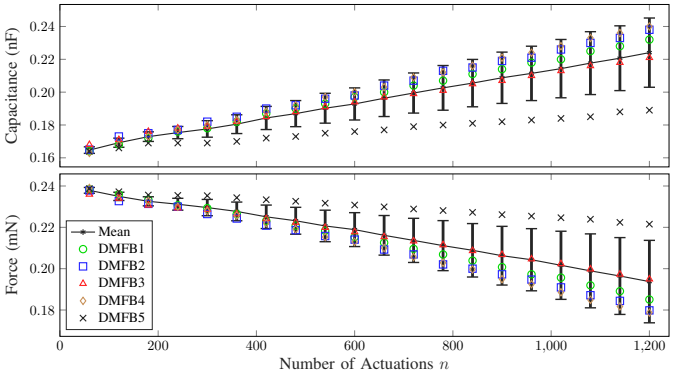


Fig. 6: Capacitance increase (top) and EWOD force degradation (bottom).

where $V_T = 130$ is threshold voltage due to soldermask insulator [40], $A(x)$ is the area of the droplet over the activated electrode, and x is the droplet position.

The degradation results, including the measured capacitance and the corresponding EWOD force, of five identical DMFBs are presented in Fig. 6. The capacitance of an electrode increases linearly as the number of actuations grows. The increase in capacitance leads to a decrease in the induced EWOD force. The coefficients of (1) are estimated as $\tau \in [0.5, 0.7]$ and $c \in [500, 800]$, which are further used in the DRL environment for agent training.

B. Evaluation

We next present the results for training agents for various configurations by showing the mean score of the agents after each training epochs for biochips of sizes $\mathsf{W} \times \mathsf{H}$, where $\mathsf{W} = \mathsf{H} \in \{30, 60, 120, 180\}$. Performance metrics consist of the mean score, the number of cycles, as well as the success rate. The metrics are collected after each training epoch by testing the agent for 500 random routing jobs, and each experiment is repeated five times. All experiments were carried

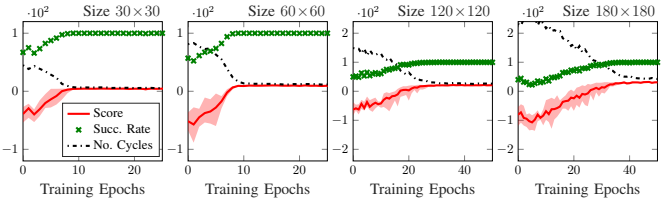


Fig. 7: Performance results for training CNNs on healthy MEDA biochips.

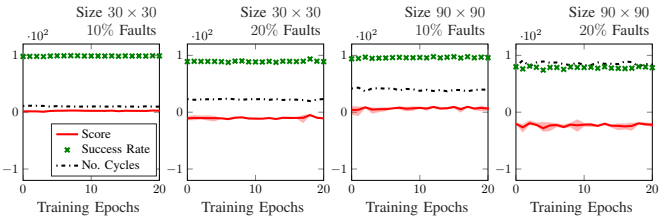


Fig. 8: Performance results for transfer learning in fault injection.

out with eight parallel environments and 2^{14} total number of steps. The training and experiments were conducted on an Intel Xeon Silver 4208 CPU and an Nvidia RTX 6000 GPU with 24 GB of memory. The training and simulation environment were implemented using Python, including OpenAI Gym and Stable-Baselines libraries.

We first trained the CNN on a healthy MEDA biochip, i.e., the number of actuations per each microelectrode is reset at the beginning of each training episode. Fig. 7 presents the CNN performance metrics versus the number of training epochs. The trends show that after a low number of epochs, a CNN learns an effective policy — i.e., the success rate converges to 100% and the score and the average number of cycles stabilize — at a relatively small number of epochs that ranges from 10 to 40 and increases with the biochip size.

We also tested the robustness of the trained agents against randomly injected faults at runtime. We used the agents trained on healthy biochips to initialize the training against biochips with randomly injected faults. Before each training episode, a fixed percentage of fully-degraded microelectrodes are randomly placed in clusters of size 2×2. Similar to the previous experiments, the trained agents were used to initialize the training on a higher percentage of faults. Fig. 8 shows the performance results for training against 10% and 20% fault injection modes. The trends demonstrate that the agents were able to adapt to the faults within the first training epoch.

Finally, to evaluate the trained CNNs, we run experiments where we compare their performance against two baselines:

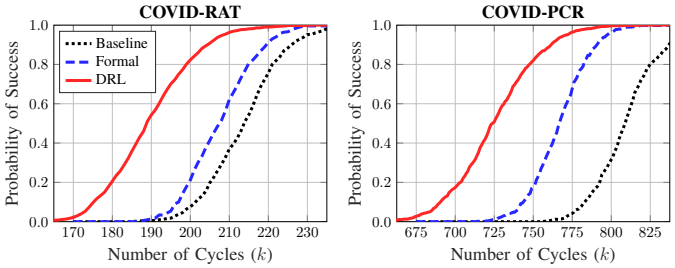


Fig. 9: Probability of successful bioassay completion vs. no. of cycles.

(i) health-agnostic policies that aim to minimize the time to reach the target without knowledge of the MC health levels (referred to as the *baseline*), and (ii) formally synthesized strategies using PRISM-games model checker (referred to as the *formal*) [19]. Each policy was used to execute two benchmark bioassays that are employed for COVID-19 testing: PCR-based (COVID-PCR) and rapid antigen-based (COVID-RAT), are widely used to detect the presence of the SARS-CoV-2 virus or the body's response to infection [35].

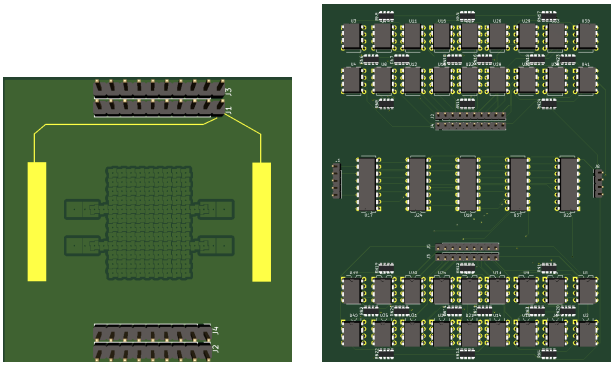
Fig. 9 presents the probability of successful bioassay completion within a given number of cycles k. The graph shows that the DRL-based routing policy outperforms the policies from the literature by achieving a significantly higher probability of success. The gain in performance is primarily due to the utilization of adaptive movement distance (see Section III). For instance, the DRL-based policy successfuly executed COVID-PCR within $k = 762$ with probability $p > 0.9$, compared to $p < 0.4$ when the other policies were used. In addition, the time needed to obtain a routing policy from the trained CNN is negligible ($t < 0.1$ sec) when compared to the formally synthesized policies where t ranged from 5 to 48 sec before each routing job.

VI. Experimental Results

In this section, we present a set of experiments whose goal is to evaluate the overall performance of the proposed DRL-based framework. Specifically, we execute routing tasks on fabricated biochips. In our experiments, we employed a PCB-based DMFB, which utilizes the same EWOD principle as MEDA biochips to manipulate droplets.

A. Environmental Setup

1) Fabricated DMFB: The DMFB is a $5.8\times5.8\,\mathrm{cm}^2$ 4-layer PCB board containing a 9×12 electrode array and four reservoirs, as shown in Fig. 10a. The size of one electrode is $1.8\times1.8\,\mathrm{mm}^2$. Every electrode is connected to one of the control pins at the sides of the DMFB. The control signals are sent from the control board through the pins to actuate the electrodes. To reduce the number of required pins and also the chip area, pin sharing is used in our DMFB design. The 9×12 electrode array is divided into three equal-sized areas, where each area includes a 9×4 electrode array. Electrodes at the corresponding position in these three arrays can be mapped to a same pin. As a result, in comparison with the control of each electrode with individual pins, only one-third of the total number of pins is needed. A layer of Cytop is coated on the PCB surface to form the hydrophobic layer on the electrodes.



(a) (b)

Fig. 10: Design of PCB boards: (a) The DMFB (b) The control board.

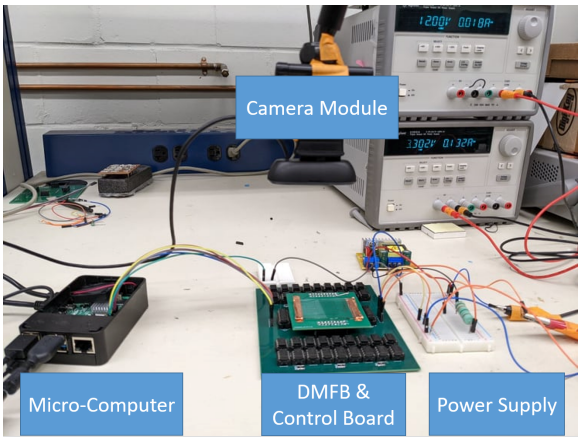


Fig. 11: The overall system used for our experiments.

2) Control Board: The control board is a $11.5\times13.5\,\mathrm{cm}^2$ 4-layer PCB design, as shown in Fig. 10b . Register ICs (P/N SN74AHC595) are used to store the actuation sequences sent from the micro-computer. At the same time, every pin that connects to the DMFB is controlled by a high voltage relay (P/N AQW210). Thus, 36 relays are needed for our DMFB design. Each relay receives the signals sent from the register ICs. Two voltage sources are applied to the control board: a source of 1 KHz and 200 V for electrode actuation, and a source of 3 V for the ICs.

3) Overall System: The overall system is shown in Fig. 11. It consists of three parts: A DMFB with a control board, a micro-computer (P/N Raspberry Pi 4), and a camera module. The routing model is deployed in the micro-computer, where an image detection program is also executed. With this image detection program, real-time images of the DMFB captured by the camera module are processed to detect the current droplet position. Based on the detection results, the micro-computer requests the next droplet action from the routing model. After receiving the output from the routing model, the micro-computer sends corresponding actuation signals to the control board to perform the desired droplet movements.

B. Routing Tasks

Fig. 12 shows the bioassay that we use in our experiments. This bioassay consists of several fluidic operations including

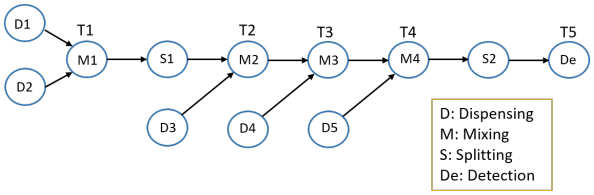


Fig. 12: The steps of the bioassay used for our experiments.

dispensing, mixing, spiltting, and detection. To focus on the routing performance of the models, we split the bioassay into five main routing tasks (T1 to T5). For each routing task, we set the maximum number of clock cycles as 40, which is around the perimeter of the electrode array. A routing task is failed and terminated when the droplet does not reach the destination after 40 steps.

C. Experimental Settings

We compare the performance of the obtained DRL-based routing policies with a baseline routing approach, which adopts the shortest-path algorithm. Both routing policies (i.e., models) are executed under two different DMFB health environments: 0% injected faults and 10% injected faults, i.e., 10% of the electrodes are degraded. The injected faults are used to simulate the aging degradation of the electrodes, which can be detected by real-time sensing. The information about the injected faults is provided as the health matrix to the DRL routing model. For each routing task, the positions of the injected faults are randomly chosen, and the electrodes that are chosen to insert faults are set to a low voltage to simulate the degradation.

In addition to injected faults, some inherent defects might also cause the failure of droplet movements; these include imperfect coating to incorporate the hydrophobic layer and PCB manufacturing defects. Unlike aging degradation, which causes changes in the capacitance and thus can be sensed in real-time, the inherent defects can not be easily sensed. Therefore, the information about these defects is not provided to the DRL routing agent. For our DMFB boards, around 5% of electrodes suffer from inherent defects. Our results show that the DRL model can ensure reliable bioassay execution even in the presence of these defects.

D. Results and Evaluation

We execute the baseline routing model and the proposed DRL routing model under two different DMFB health environments with five routing tasks (T1 to T5). Fig. 13(top) shows the average number of clock cycles needed for each routing task. Under the environment with 0% injected faults, the developed DRL policy can achieve a similar performance as the baseline that adopts the shortest path. When the injected fault rate is 10%, our results show that the DRL model needs fewer clock cycles than the baseline approach. Similar results can be seen in Fig. 13(bottom), where the average execution time for each operation is shown. The execution time for a routing task includes the computation time of the model and the electrode actuation time.

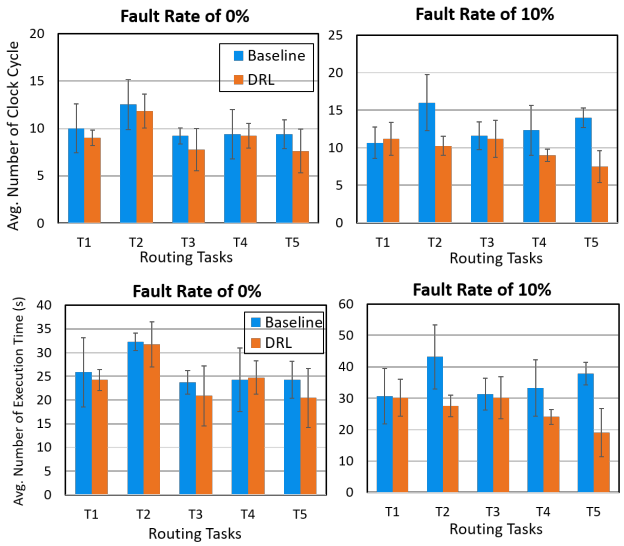


Fig. 13: Average number of clock cycles and average execution time for each routing task under fault rate of 0% and 10%. When the fault rate is 10%, 10 electrodes on the DMFB are chosen as degraded and set to a low voltage: (top) Average number of clock cycles for each routing task; (bottom) Average execution time for each routing task.

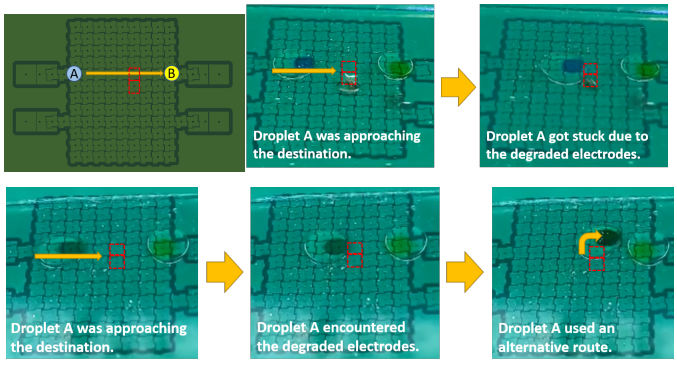


Fig. 14: Execution the routing task T1 with degraded electrodes: (top-left) The intended routing task T1, (top-right) the routing progression for the baseline model, (bottom) the routing progression for the DRL model.

Fig. 14 shows an example when the baseline approach and the DRL-based routing policy execute a routing task with degraded electrodes on the routing path. Fig. 14(top-left) shows the routing task T1, which performs the mixing of droplet A and droplet B. Thus, we transport droplet A to the location corresponding to droplet B. The two degraded electrodes are marked with red squares. Fig. 14(top-right) shows routing progression for the baseline model. The droplet followed the shortest path to approach the destination before encountering the degraded electrodes. When the droplet encountered the degraded electrodes, it got stuck at the same position until the maximum number of steps was reached and the assay was aborted. On the other hand, Fig. 14(bottom) shows routing progression for the DRL model. When the droplet encountered the degraded electrodes, the DRL-based policy chose an alternative route and reached the destination successfully. The video of this example can be found in [41].

Consequently, our experiments show that the proposed DRL-based routing approach can be effectively integrated into

a DMFB system. The experimental results show that the DRL-based routing policy provides reliable routing results even in the presence of degraded electrodes.

VII. Conclusion

We have presented a deep reinforcement learning (DRL) framework that can respond to microelectrode degradation during droplet routing on MEDA biochips. Our framework adopts proactive health monitoring on individual microelectrodes and uses the captured response to plan more feasible route for droplet transportation. We have shown that the bioassay execution time and the number of clock cycles are significantly reduced when our approach is employed. Our results also show that the DRL-based routing policies facilitate real-time adaptation to faulty microelectrodes as MCs degrade over time.

References

[1] W.-L. Chou, P.-Y. Lee, C.-L. Yang, W.-Y. Huang, and Y.-S. Lin, "Recent advances in applications of droplet microfluidics," *Micromachines*, vol. 6, no. 9, pp. 1249–1271, 2015.

[2] M. Ibrahim, C. Boswell, K. Chakrabarty, K. Scott, and M. Pajic, "A real-time digital-microfluidic platform for epigenetics," in *2016 International Conference on Compliers, Architectures, and Sythesis of Embedded Systems (CASES)*, Oct 2016, pp. 1–10.

[3] R. S. Sista, R. Ng, M. Nuffer, M. Basmajian, J. Coyne, J. Elderbroom, D. Hull, K. Kay, M. Krishnamurthy, C. Roberts, D. Wu, A. D. Kennedy, R. Singh, V. Srinivasan, and V. K. Pamula, "Digital microfluidic platform to maximize diagnostic tests with low sample volumes from newborns and pediatric patients," *Diagnostics*, vol. 10, no. 1, 2020. [Online]. Available: https://www.mdpi.com/2075-4418/10/1/21

[4] S. Huang, J. Connolly, A. Khlystov, and R. B. Fair, "Digital microfluidics for the detection of selected inorganic ions in aerosols," *Sensors*, vol. 20, no. 5, p. 1281, 2020.

[5] A. Ganguli, A. Mostafa, J. Berger, M. Y. Aydin, F. Sun, S. A. S. d. Ramirez, E. Valera, B. T. Cunningham, W. P. King, and R. Bashir, *Proceedings of the National Academy of Sciences*, vol. 117, no. 37, pp. 22 727–22 735, 2020.

[6] C. Sheridan, "COVID-19 spurs wave of innovative diagnostics," *Nature biotechnology*, vol. 38, no. 7, pp. 769–772, 2020.

[7] S. Poddar, S. Ghoshal, K. Chakrabarty, and B. B. Bhattacharya, "Error-correcting sample preparation with cyberphysical digital microfluidic lab-on-chip," *ACM Transactions on Design Automation of Electronic Systems*, vol. 22, no. 1, pp. 1–29, 2016.

[8] T.-C. Liang, Z. Zhong, M. Pajic, and K. Chakrabarty, "Extending the lifetime of MEDA biochips by selective sensing on microelectrodes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3531–3543, 2020.

[9] K. Y.-T. Lai, Y.-T. Yang, and C.-Y. Lee, "An intelligent digital microfluidic processor for biomedical detection," *Journal of Signal Processing Systems*, vol. 78, no. 1, pp. 85–93, 2015.

[10] Y. Ho, G. Wang, K. Y.-T. Lai, Y.-W. Lu, K.-M. Liu, Y.-M. Wang, and C.-Y. Lee, "Design of a micro-electrode cell for programmable lab-on-cmos platform," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 2871–2874.

[11] Y. Zhao, T. Xu, and K. Chakrabarty, "Integrated control-path design and error recovery in the synthesis of digital microfluidic lab-on-chip," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 6, no. 3, 2010.

[12] Z. Zhong, Z. Li, and K. Chakrabarty, "Adaptive and roll-forward error recovery in MEDA biochips based on droplet-aliquot operations and predictive analysis," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, pp. 577–592, 2018.

[13] Z. Li, K. Y.-T. Lai, P.-H. Yu, K. Chakrabarty, M. Pajic, T.-Y. Ho, and C.-Y. Lee, "Error recovery in a micro-electrode-dot-array digital microfluidic biochip," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–8.

[14] Z. Li, K. Y. Lai, J. McCrone, P. Yu, K. Chakrabarty, M. Pajic, T. Ho, and C. Lee, "Efficient and adaptive error recovery in a micro-electrode-dot-array digital microfluidic biochip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 3, pp. 601–614, March 2018.

[15] M. Elfar, Z. Zhong, Z. Li, K. Chakrabarty, and M. Pajic, "Synthesis of error-recovery protocols for micro-electrode-dot-array digital microfluidic biochips," *ACM Transactions on Embedded Computing Systems*, vol. 16, no. 5s, pp. 127:1–127:22, Sep. 2017. [Online]. Available: http://doi.acm.org/10.1145/3126538

[16] T. C. Liang, Z. Zhong, M. Pajic, and K. Chakrabarty, "Extending the lifetime of meda biochips by selective sensing on microelectrodes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3531–3543, 2020.

[17] T.-C. Liang, Z. Zhong, Y. Bigdeli, T.-Y. Ho, K. Chakrabarty, and R. Fair, "Adaptive droplet routing in digital microfluidic biochips using deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[18] J.-i. Yoshida, A. Nagaki, and T. Yamada, "Flash chemistry: fast chemical synthesis by using microreactors," *Chemistry–A European Journal*, vol. 14, no. 25, pp. 7450–7459, 2008.

[19] M. Elfar, T.-C. Liang, K. Chakrabarty, and M. Pajic, "Formal synthesis of adaptive droplet routing for MEDA biochips," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*. IEEE, 2021.

[20] ——, "Formal synthesis of adaptive droplet routing for meda biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[21] S. Anderson, B. Hadwen, and C. Brown, "Thin-film-transistor digital microfluidics for high value in vitro diagnostics at the point of need," *Lab on a Chip*, vol. 21, no. 5, pp. 962–975, 2021.

[22] M. Elfar, T.-C. Liang, K. Chakrabarty, and M. Pajic, "Adaptive droplet routing for meda biochips via deep reinforcement learning," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2022.

[23] C. Quilliet and B. Berge, "Electrowetting: a recent outbreak," *Current Opinion in Colloid & Interface Science*, vol. 6, no. 1, pp. 34–39, 2001.

[24] "illumina Official Website," https://www.illumina.com/techniques/sequencing/ngs-library-prep/automation.html, 2022 [online].

[25] "Genmark Official Website," https://www.genmarkdx.com, 2022 [online].

[26] "Baebies Official Website," https://baebies.com, 2022 [online].

[27] F. Su, S. Ozev, and K. Chakrabarty, "Ensuring the operational health of droplet-based microelectrofluidic biosensor systems," *IEEE Sensors Journal*, vol. 5, no. 4, pp. 763–773, 2005.

[28] Z. Zhong, T.-C. Liang, and K. Chakrabarty, "Enhancing the reliability of meda biochips using ijtag and wear leveling," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 10, pp. 2063–2076, 2021.

[29] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3389–3396.

[30] H.-A. M. C. e. a. Silver, D., "Mastering the game of go with deep neural networks and tree search," in *Nature 529*, 2016, p. 484–489.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[32] Q. Gao, M. Naumann, I. Jovanov, V. Lesi, K. Kamaravelu, W. M. Grill, and M. Pajic, "Model-based design of closed loop deep brain stimulation controller using reinforcement learning," in *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*, 2020, pp. 108–118.

[33] H. Verheijen and M. Prins, "Reversible electrowetting and trapping of charge: model and experiments," *Langmuir*, vol. 15, pp. 6616–6620, 1999.

[34] S. D. Whitehead and D. H. Ballard, "Learning to perceive and act by trial and error," *Machine Learning*, vol. 7, no. 1, pp. 45–83, 1991.

[35] G. Guglielmi, "Fast coronavirus tests: what they can and can't do." *Nature*, vol. 585, no. 7826, pp. 496–498, 2020.

[36] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2016, pp. 1928–1937.

[37] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[38] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
[39] Z. Li, K. Y.-T. Lai, P.-H. Yu, K. Chakrabarty, T.-Y. Ho, and C.-Y. Lee, "Droplet size-aware high-level synthesis for micro-electrode-dot-array digital microfluidic biochips," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 3, pp. 612–626, 2017.
[40] R. B. Fair, "Digital microfluidics: is a true lab-on-a-chip possible?" *Microfluidics and Nanofluidics*, vol. 3, no. 3, pp. 245–281, 2007.
[41] M. Elfar, Y.-C. Chang H. H.-Y. Ku, T.-C. Liang, K. Chakrabarty, and M. Pajic, "Recorded video for the routing example." https://reurl.cc/AK20pZ, 2022 [online].