2025 4th International Conference on Artificial Intelligence, Internet of Things and Cloud Computing Technology (AIoTC)

2025 4th International Conference on Artificial Intelligence, Internet of Things and Cloud Computing Technology (AIoTC) | 979-8-3315-9720-7/25/$31.00 ©2025 IEEE | DOI: 10.1109/AIOTC66747.2025.11198609

Creating Personalized Medicine ML Models Using Distributed Learning Via Ray

Chaeeun Kim, William Self, Nick Wang, Daniel Wen, Chen Qian*, Liting Hu*
University of California Santa Cruz
{ckim151, wself, nijwang, dywen, cqian12, liting}@ucsc.edu

***Abstract*—Sepsis is a life-threatening condition that requires timely and personalized treatment in intensive care units. Recent studies have applied reinforcement learning, or RL, to model optimal treatment policies, but traditional single-threaded training approaches are computationally expensive, requiring hundreds of thousands of episodes to converge. In this paper, we propose a distributed RL framework for sepsis treatment that parallelizes agent training using Ray, enabling scalable and efficient learning. We implement three RL algorithms: DQN, SARSA, and SAC, and evaluate them on the ICU-Sepsis environment. Our method significantly reduces training time and sample complexity while maintaining policy accuracy. Specifically, our Ray-based approach achieves up to 40× fewer total episodes and up to 125× faster convergence, without compromising average reward. These results demonstrate that distributed RL can accelerate the development of clinically meaningful policies in personalized medicine.**

***Index Terms*—Reinforcement Learning, Distributed Machine Learning, Ray, Personalized Medicine, Healthcare AI, Cloud-based Training, Sepsis Detection**

I. Introduction

The convergence of artificial intelligence [1], cloud computing [2], [3], and healthcare [4] has opened new avenues for developing intelligent, adaptive, and personalized medical systems. In particular, personalized medicine, which tailors treatment strategies to individual patients based on genetic, physiological, and lifestyle factors, has become a critical area of innovation [5]. However, building robust and accurate AI models in this domain presents several challenges, including limited access to high-quality medical datasets and the computational cost of training complex models [6].

Reinforcement Learning (RL) [7] has emerged as a powerful paradigm for modeling sequential decision-making tasks in healthcare, such as treatment planning [8] and intervention strategies [9]. Yet, training RL agents, especially in simulated medical environments like ICU-Sepsis, is computationally intensive and often slow to converge. These limitations hinder the deployment of RL-based decision systems in real-world clinical settings where timeliness and efficiency are paramount.

To address this challenge, we investigate the use of Ray [10], a high-performance distributed computing framework, to accelerate the training of RL algorithms in the ICU-Sepsis environment [11]. By leveraging Ray's parallelism and scalability, we aim to significantly reduce training time while preserving or even enhancing the quality of learned policies. We focus on three RL algorithms: Deep Q-Network (DQN) [12], State-Action-Reward-State-Action (SARSA) [13], and Soft Actor-Critic (SAC) [14], and demonstrate how distributed training can yield substantial speedups with no degradation in clinical performance metrics.

This work contributes to the growing body of research at the intersection of AI, cloud computing, and healthcare, and illustrates how scalable, distributed systems can empower intelligent medical decision-making. Our evaluations show up to 40× faster convergence in RL training, suggesting that cloud-enabled AI frameworks like Ray are well-suited for time-sensitive applications in personalized medicine.

* Corresponding authors.

II. Background and Challenge

This section introduces the key concepts underlying our work, including personalized medicine, AI in healthcare, reinforcement learning for treatment planning, and distributed training. We also highlight the core challenges that motivate our approach.

A. Personalized Medicine

Personalized medicine seeks to tailor diagnosis and treatment strategies to individual patients based on genetic, environmental, and lifestyle factors. This contrasts with traditional "one-size-fits-all" approaches [15], which apply generalized protocols to populations with similar symptoms. By leveraging fine-grained patient data, personalized approaches can improve diagnostic accuracy and therapeutic outcomes.

Recent progress in genome sequencing technologies, such as next-generation sequencing (NGS) [16], whole-genome sequencing (WGS) [17], and whole-exome sequencing (WES) [18], has accelerated the development of precision diagnostics across a range of medical conditions, including cancer and rare genetic disorders. However, widespread adoption remains limited by challenges in data interpretation, privacy, integration into clinical workflows, and the need for clinician training in genomic literacy.

Moreover, incorporating patient-specific traits into machine learning models requires methods that can generalize across heterogeneous modalities while remaining sensitive to individual variations. Efforts to build invariant yet personalized representations in medical imaging [19] have shown promise, but generalization and robustness remain open research problems.

B. AI in Medicine

Artificial intelligence is increasingly used to support personalized healthcare. Supervised and unsupervised learning models have been developed to predict patient risk, recommend treatments, and analyze diagnostic images. In personalized oncology, for instance, machine learning has been applied to high-throughput screening (HTS) data to predict drug responses in patient-derived cell lines [20]. These models can identify effective treatments while reducing the need for exhaustive lab testing.

However, medical AI also faces significant challenges. Model performance is often sensitive to bias in training data, especially in observational datasets. Studies have shown that not all biases negatively impact outcomes; some can even be leveraged to improve reliability. Nevertheless, identifying and controlling for harmful biases remains an active area of research [21]. Furthermore, translating predictive insights into clinically actionable decisions requires interpretability, regulatory approval, and strong real-world validation.

C. Reinforcement Learning

Reinforcement learning (RL) [7] offers a natural framework for sequential decision-making, where agents learn to maximize cumulative reward by interacting with an environment. In healthcare, RL has been applied to dynamic treatment planning and dosage adjustment [8]. The ICU-Sepsis environment [11] is a widely used benchmark based on the MIMIC-III dataset [22], simulating patient trajectories and outcomes in intensive care.

Although RL provides powerful tools for optimizing treatment policies, its application to medical settings is constrained by long training times and high sample complexity. Most RL algorithms [23] require hundreds of thousands of episodes to converge, which makes exploration and hyperparameter tuning prohibitively slow in high-fidelity simulations like ICU-Sepsis.

D. Distributed Learning

To accelerate model training, distributed computing frameworks such as Ray [10] have become increasingly popular. Ray enables scalable parallelism across CPUs and GPUs, making it well-suited for RL workloads where multiple agents can collect experience in parallel and share it through centralized learners.

Distributed learning offers clear benefits in training speed and throughput, but also introduces overhead due to interprocess communication, data transfer, and coordination. These bottlenecks must be carefully managed to avoid offsetting the gains from parallelization. Understanding and mitigating this trade-off is essential when deploying distributed RL systems in time-sensitive domains like critical care.

E. Challenges in Distributed Learning for Personalized Medicine

From the above, we identify several core challenges:

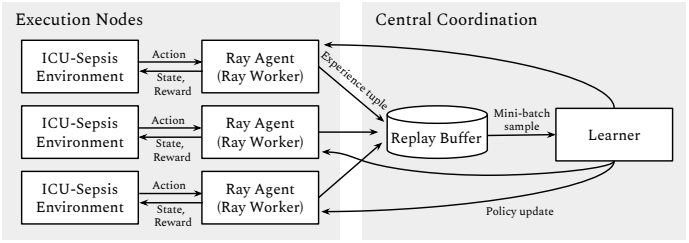

Fig. 1. Overall system architecture for distributed reinforcement learning with Ray.

- **Computational Inefficiency in Reinforcement Learning:** Training RL agents in clinical simulation environments like ICU-Sepsis requires a large number of episodes to converge. Standard algorithms such as DQN [12], SARSA [13], or SAC [14] can take hundreds of thousands of episodes to learn meaningful policies. This results in long training cycles and limits experimentation with different model architectures or reward formulations.
- **Scalability and System-Level Bottlenecks:** While distributed computing frameworks such as Ray offer the potential to parallelize RL training, realizing those benefits at scale is non-trivial. Challenges include coordinating communication among workers, managing shared memory efficiently, and minimizing overhead from distributed scheduling. Improper parallelization can result in diminishing returns or even performance degradation.
- **Data Limitations and Clinical Constraints:** In real-world healthcare applications, collecting high-quality, diverse, and ethically sourced patient data is difficult. This is particularly true for life-threatening conditions like sepsis, where interventions are time-sensitive and experimental data is scarce. Effective systems must therefore rely on simulation-based environments while still generalizing to the complexities of real-world clinical settings.

Our work is designed to address these challenges by integrating distributed reinforcement learning via Ray in a clinically inspired simulation environment, enabling faster and more scalable training of personalized treatment policies.

III. System Design

This section describes the distributed reinforcement learning system we implemented using the Ray framework to accelerate agent training in the ICU-Sepsis environment. Our system is designed to scale across multiple CPU cores, enabling parallel experience collection and faster convergence for various RL algorithms. We explain the architecture, components, and the integration of Ray with the ICU-Sepsis simulator.

A. Overview

Fig. 1 presents the high-level architecture of our system. The architecture consists of multiple Ray Agents (workers) that interact with independent instances of the ICU-Sepsis

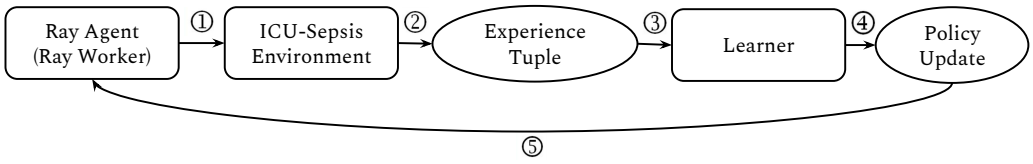

Fig. 2. Execution Workflow for Distributed Reinforcement Learning

Algorithm 1 Deep Q-Network (DQN)

Initialize Q-network and target Q-network with random weights
Initialize replay buffer $\mathcal{D}$
for each episode **do**
  $s \leftarrow$ env.reset()
  for each step **do**
    Select action a using ϵ-greedy policy based on $Q(s, \cdot)$
    Execute action a, observe r, $s' \leftarrow$ env.step(a)
    Store transition (s, a, r, s') in $\mathcal{D}$
    Sample mini-batch from $\mathcal{D}$
    Compute target: $y \leftarrow r + \gamma \max_{a'} Q_{\text{target}}(s', a')$
    Update Q-network by minimizing loss: $(Q(s, a) - y)^2$
    Periodically update $Q_{\text{target}} \leftarrow Q$
  end for
end for

Algorithm 2 SARSA Training Algorithm

Initialize Q-network with random weights
for each episode **do**
  $s \leftarrow$ env.reset()
  Choose action a using ϵ-greedy policy
  for each step **do**
    Take action a, observe r, s'
    Choose next action a' using ϵ-greedy policy
    Compute target: $y \leftarrow r + \gamma Q(s', a')$
    Update Q-network to minimize loss: $(Q(s, a) - y)^2$
    $s \leftarrow s'$, $a \leftarrow a'$
  end for
end for

Algorithm 3 Soft Actor-Critic (SAC) Training Algorithm

Initialize policy network π, Q-networks Q_1, Q_2, target Q-networks
Initialize replay buffer $\mathcal{D}$
for each iteration **do**
  Collect data using current policy π
  Store transition (s, a, r, s') into $\mathcal{D}$
  Sample mini-batch from $\mathcal{D}$
  Compute target:

$$y \leftarrow r + \gamma \left(\min_{i=1,2} Q_i^{\text{target}}(s', a') - \alpha \log \pi(a'|s') \right)$$

  Update Q-networks by minimizing $(Q_i(s, a) - y)^2$
  Update policy network π via gradient ascent on expected reward + entropy
  Periodically update target Q-networks
end for

environment. These agents collect experience tuples and store them in a centralized Replay Buffer, which is shared via Ray's distributed object store. A Learner module samples from this buffer to perform gradient updates and periodically sends updated policy weights back to the agents.

This distributed execution setup allows parallel environment interactions while maintaining centralized, synchronized learning. As a result, the system improves data throughput and accelerates the convergence of training.

B. Ray Agent

Each Ray Agent is a lightweight worker process responsible for interacting with an independent instance of the ICU-Sepsis environment. At every time step, the agent selects an action based on the current policy, which may be stochastic (SAC) or deterministic (DQN, SARSA), and executes this action in the environment. The agent then observes the next state and reward and constructs an experience tuple (s, a, r, s').

These tuples are forwarded to a centralized learner, either directly or through the shared Replay Buffer, depending on the algorithm. Ray Agents periodically receive updated policy weights from the Learner, allowing them to act using the most recent policy. This decoupling of experience collection from learning enables high parallelism and scalability, as multiple agents can operate independently and asynchronously.

C. ICU-Sepsis Environment

The ICU-Sepsis environment simulates a medical decision-making task based on the MIMIC-III dataset. The environment defines a discrete-time Markov Decision Process (MDP), where:

- The state includes clinical features such as vital signs, lab measurements, and demographic information.
- The action space corresponds to medical interventions like administering vasopressors or intravenous fluids.
- The reward function is sparse and binary: agents receive +1 for survival and -1 for death at the end of an episode.

Each agent interacts with an instance of this environment by choosing actions at each time step. This design mimics real ICU treatment scenarios while maintaining a lightweight, tabular structure suitable for training RL agents.

D. Replay Buffer

The Replay Buffer is a shared memory structure used to store experience tuples (s, a, r, s') collected by the Ray Agents. It enables the system to reuse past experiences for learning, which improves sample efficiency and stabilizes training by breaking correlations between consecutive samples.

For off-policy algorithms like DQN and SAC, the Learner continuously samples mini-batches from this buffer to compute loss and perform parameter updates. The buffer is implemented using Ray's distributed object store, allowing efficient data sharing across processes and nodes. For SARSA, which is an on-policy algorithm, the buffer plays a reduced role since learning relies more on current trajectories. Nonetheless, maintaining a centralized buffer ensures compatibility across all algorithms.

E. Learner

The Learner is the central component that performs model optimization using the experiences provided by Ray Agents. It runs continuously in the background and periodically samples mini-batches from the Replay Buffer. Depending on the selected RL algorithm, the Learner computes value function updates (DQN, SARSA) or actor-critic gradients (SAC) to refine the policy and/or Q-functions.

The updated weights are pushed back to the Ray Agents at regular intervals, ensuring policy synchronization across the system. By separating data collection from policy updates, the Learner architecture enables stable and scalable training, especially when working with multiple parallel environments.

F. Reinforcement Learning Algorithms

This system supports three representative reinforcement learning algorithms: Deep Q-Network (DQN), SARSA, and Soft Actor-Critic (SAC). Each algorithm follows a distinct learning paradigm, such as off-policy versus on-policy and value-based versus actor-critic, offering flexibility to adapt to various environments and learning speeds.

1) Deep Q-Network: Deep Q-Network (DQN) approximates the optimal action-value function $Q^*(s, a)$ using a deep neural network. It updates the Q-values based on the Bellman equation:

$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right]$

To enhance stability and efficiency, DQN uses two key techniques:

- **Experience Replay:** Stores past transitions in a replay buffer and samples mini-batches uniformly to reduce correlation.
- **Target Network:** Maintains a separate target Q-network Q_{target} that is periodically synchronized with the main network.

The step-by-step learning process of DQN is shown in Algorithm 1.

2) SARSA: SARSA is an on-policy value-based algorithm. It updates Q-values using the actual action taken by the current policy rather than the greedy one. The update rule is:

$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma Q(s', a') - Q(s, a)\right]$

Because SARSA follows the current policy, it tends to produce safer and more conservative policies, which is especially valuable in clinical applications. The detailed training process for SARSA is described in Algorithm 2.

3) Soft Actor-Critic: Soft Actor-Critic (SAC) is an off-policy actor-critic method that optimizes both the expected reward and the policy's entropy. The objective function is:

$J(\pi) = \sum_t \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} \left[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))\right]$

SAC includes several components that enable robust learning:

- **Stochastic Actor:** Learns a distribution over actions, enabling better exploration.
- **Twin Critics:** Mitigates overestimation bias by learning two Q-functions and using the minimum.
- **Entropy Regularization:** Encourages high-entropy policies for improved robustness.

SAC is particularly effective in continuous control tasks and complex environments. Its training process is summarized in Algorithm 3.

G. Execution Workflow

Fig. 2 illustrates the end-to-end workflow of our distributed reinforcement learning system powered by Ray. (1) The process begins with each Ray Agent (Ray Worker) selecting an action based on its current policy. (2) This action is executed in the ICU-Sepsis Environment, which returns the resulting state and reward, simulating the patient's physiological response to the treatment. (3) The agent then forms an Experience Tuple, which is a structured representation of the state, action, reward, and next state. This tuple is forwarded to the Learner. (4) The Learner uses this data to compute gradients and update the model policy according to the selected RL algorithm, such as DQN, SARSA, or SAC. (5) The updated policy weights are sent back to the Ray Agent, completing one full training loop. This process repeats continuously and asynchronously to enable distributed learning across multiple agents and achieve more efficient convergence.

IV. Evaluation

This section presents a comprehensive evaluation of our distributed reinforcement learning system in the ICU-Sepsis environment. We assess both clinical performance and computational efficiency across multiple reinforcement learning algorithms and system configurations. Our goal is to demonstrate that the proposed design not only improves convergence time and scalability but also maintains strong clinical relevance in policy quality.

A. Evaluation Metrics

We use two primary metrics from the ICU-Sepsis benchmark: (1) average return achieved by each RL algorithm, and (2) the number of episodes until convergence. These metrics

TABLE I
COMPARISON OF RL ALGORITHMS BEFORE AND AFTER RAY-BASED PARALLELIZATION.

Metric	DQN	SARSA	SAC	DQN w/ Ray	SARSA w/ Ray	SAC w/ Ray
Total Episodes	500k	500k	500k	∼13.5k	15.6k	∼12.5k
Episodes to Converge	∼450–500k	∼450–500k	∼400k	∼6k	∼4k	∼11.5k
Avg. Reward	0.79	0.75	0.77	0.78	0.78	0.77

reflect how effectively a learned policy maximizes patient survival in the simulated ICU setting.

To assess scalability and efficiency, we also measure (3) total training time and (4) system-level resource usage. These metrics are recorded for both our distributed Ray-based system and a baseline single-machine implementation.

B. Experimental Setup

The baseline single-machine system uses an AMD Ryzen 9 7950X 16-core CPU with hyperthreading (32 threads total) and an NVIDIA GeForce RTX 3090 GPU. This configuration allows us to fairly compare performance with our Ray-based distributed training system across identical hardware.

In addition, we conduct a comparative study of different reinforcement learning algorithms (DQN, SARSA, SAC) under various hyperparameter settings and reward structures. This analysis helps identify the most effective configurations in terms of both policy quality and computational cost.

C. Impact of Distributed Training on Convergence and Sample Efficiency

To evaluate the benefits of distributed reinforcement learning, we compare baseline single-threaded training with our Ray-based parallel implementation for three algorithms: DQN, SARSA, and SAC, all trained in the ICU-Sepsis environment. Rather than visualizing learning curves, we summarize key metrics in Table I, including total episodes, episodes to convergence, and final average reward.

1) Convergence Acceleration through Parallelism: In the baseline configuration, each RL agent required roughly 500,000 episodes to converge. When applying Ray to utilize parallel actors across multiple CPU cores, we observe dramatic reductions in both total episodes and episodes to convergence. Specifically, DQN completes training with approximately 13,500 total episodes and converges after just 6,000 episodes, which represents an 83× improvement in convergence efficiency. Similarly, SARSA converges after 4,000 episodes, achieving 125× faster convergence, and SAC converges after 11,500 episodes, which is roughly 43× faster.

These improvements are achieved without sacrificing policy quality. The average rewards remain consistent with the baseline: 0.78–0.79 for DQN, 0.78 for SARSA, and 0.77 for SAC. This suggests that distributed training can significantly accelerate learning while maintaining comparable model performance.

2) Training Efficiency and Real-World Implications: Such training speedups have major implications for time-sensitive applications like ICU treatment planning. Faster training enables the development of real-time personalized policies for new patients or updated models based on evolving patient data. Additionally, our Ray-based setup leverages shared resources efficiently, reducing training time while ensuring scalability to larger environments or higher-resolution patient data.

3) Limitations and System Overheads: Despite these gains, distributed training introduces coordination and communication overhead. Each Ray worker periodically synchronizes with the Global Control Store (GCS), and redundant computation across actors adds some inefficiency. Moreover, as we scale up to more cores or move to a distributed GPU cluster, latency and communication bottlenecks may become more significant. These trade-offs must be carefully managed, especially in large-scale deployments.

4) Takeaways: Our results demonstrate that distributed RL using Ray is highly effective in reducing training time without compromising performance. While some overhead exists, the net gain in convergence efficiency outweighs these costs. These findings suggest that distributed RL frameworks are a promising tool for accelerating personalized healthcare models, especially in time-critical scenarios such as sepsis treatment.

V. CONCLUSION

In this paper, we propose a Ray-based distributed reinforcement learning system for training personalized medicine models on ICU-Sepsis data. Our key contribution is the parallelization of RL training using Ray actors, which significantly improves convergence speed and sample efficiency without compromising model performance. Unlike prior work that focuses solely on algorithmic performance, our approach emphasizes scalable training infrastructure and evaluates multiple RL algorithms (DQN, SARSA, SAC) under various hyperparameter and reward settings. Our evaluation shows that distributed training reduces total training episodes by over 30× and episodes to convergence by up to 125×, while maintaining comparable average rewards to single-threaded baselines. These results demonstrate the potential of distributed RL to accelerate model development in time-critical healthcare domains.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grants NSF CAREER-2313737, NSF OAC-2313738, and NSF CNS-2322919.

REFERENCES

[1] I. Ghebrehiwet, N. Zaki, R. Damseh, and M. S. Mohamad, *Revolutionizing personalized medicine with generative AI: a systematic review*. Springer, 2024.

[2] S. S. Vellela, B. V. Reddy, K. K. Chaitanya, and M. V. Rao, "An integrated approach to improve e-healthcare system using dynamic cloud computing platform," in *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 2023.

[3] C. Kim, C. Han, and H.-M. Park, "Bts: Load-balanced distributed union-find for finding connected components with balanced tree structures," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 2024.

[4] D. Saraswat, P. Bhattacharya, A. Verma, V. K. Prasad, S. Tanwar, G. Sharma, P. N. Bokoro, and R. Sharma, "Explainable ai for healthcare 5.0: opportunities and challenges," *IEEe Access*, 2022.

[5] S. Vadapalli, H. Abdelhalim, S. Zeeshan, and Z. Ahmed, "Artificial intelligence and machine learning approaches using gene expression and variant data for personalized medicine," *Briefings in bioinformatics*, 2022.

[6] A. A. Abdellatif, N. Mhaisen, A. Mohamed, A. Erbad, and M. Guizani, "Reinforcement learning for intelligent healthcare systems: A review of challenges, applications, and open research issues," *IEEE Internet of Things Journal*, 2023.

[7] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, 1996.

[8] X. Wu, R. Li, Z. He, T. Yu, and C. Cheng, "A value-based deep reinforcement learning model with human expertise in optimal treatment of sepsis," *NPJ Digital Medicine*, 2023.

[9] H. Liu, K. Cai, P. Li, C. Qian, P. Zhao, and X. Wu, "Redrl: A review-enhanced deep reinforcement learning model for interactive recommendation," *Expert Systems with Applications*, 2023.

[10] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, W. Paul, M. I. Jordan, and I. Stoica, "Ray: A distributed framework for emerging AI applications," *CoRR*, 2017.

[11] K. Choudhary, D. Gupta, and P. S. Thomas, "Icu-sepsis: A benchmark mdp built from real medical data," 2024.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nat.*, 2015.

[13] G. A. Rummery and M. Niranjan, "On-line q-learning using connectionist systems," 1994.

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 2018.

[15] F. S. Collins and H. Varmus, "A new initiative on precision medicine," *New England Journal of Medicine*, 2015.

[16] J. M. Heather and B. Chain, "The sequence of sequencers: The history of sequencing dna," *Genomics*, 2016.

[17] F. E. Dewey, M. E. Grove, C. Pan, B. A. Goldstein, J. A. Bernstein, H. Chaib, J. D. Merker, R. L. Goldfeder, G. M. Enns, S. P. David *et al.*, "Clinical interpretation and implications of whole-genome sequencing," *Jama*, 2014.

[18] Y. Yang, D. M. Muzny, J. G. Reid, M. N. Bainbridge, A. Willis, P. A. Ward, A. Braxton, J. Beuten, F. Xia, Z. Niu *et al.*, "Clinical whole-exome sequencing for the diagnosis of mendelian disorders," *New England Journal of Medicine*, 2013.

[19] M. Tsuneki, "Deep learning models in medical image analysis," *Journal of Oral Biosciences*, 2022.

[20] J. C. Costello, L. M. Heiser, E. Georgii, M. Gönen, M. P. Menden, N. J. Wang, M. Bansal, M. Ammad-Ud-Din, P. Hintsanen, S. A. Khan *et al.*, "A community effort to assess and improve drug sensitivity prediction algorithms," *Nature biotechnology*, 2014.

[21] R. J. Chen, J. J. Wang, D. F. Williamson, T. Y. Chen, J. Lipkova, M. Y. Lu, S. Sahai, and F. Mahmood, "Algorithmic fairness in artificial intelligence for medicine and healthcare," *Nature biomedical engineering*, 2023.

[22] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, 2016.

[23] N. Cheng, X. Wang, Z. Li, Z. Yin, T. Luan, and X. S. Shen, "Toward enhanced reinforcement learning-based resource management via digital twin: Opportunities, applications, and challenges," *IEEE Network*, 2024.