Optimizing Reinforcement Learning Using Failure Data

Suzeyu (George) Cui
Livingston High School
Livingston, USA
george.suzeyu.cui@gmail.com

Jesse Parron
School of Computing
Montclair State University
Montclair, USA
parronj1@montclair.edu

Garrett Modery
School of Computing
Montclair State University
Montclair, USA
moderyg1@motclair.edu

Weitian Wang
School of Computing
Montclair State University
Montclair, USA
wangw@montclair.edu

2024 IEEE MIT Undergraduate Research Technology Conference (URTC) | 979-8-3315-3100-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/URTC65039.2024.10937604

***Abstract*— Learning from both successes and failures is key to developing robust and efficient policies in reinforcement learning (RL). Traditional RL excels at learning from rewards but often neglects non-rewarding states, especially those leading to negative outcomes. This paper introduces a novel approach that integrates a modified Gaussian distribution into a Deep Q-Network (DQN) framework to learn from failures. By penalizing state-action pairs near historical failure points, the model guides the agent away from pitfalls. The optimized DQN shows improved learning speed and stability, achieving higher and more consistent scores than a standard DQN. This approach highlights the potential of hybrid RL models that combine value-based methods with failure-aware mechanisms to accelerate learning and enhance decision-making.**

***Keywords*—robotics, reinforcement learning, machine learning, learning from failure**

I. INTRODUCTION

Reinforcement learning (RL), despite its advances, often relies on brute-force exploration to discover optimal strategies. This approach, though effective in simulations where multiple scenarios can be tested in fractions of a second, still has some gaps in real-world applications [1]. In scenarios involving robotics and industrial machinery, training RL models can be time-consuming and often requires supervision, leading to increased costs and extended development cycles [2]. This inefficiency highlights the need for more sophisticated methods that enable machines to learn from past experiences, particularly failures, to accelerate and refine the learning process. In a world increasingly driven by data and digital interactions, the ability to learn from failures and adapt accordingly is a necessity for the advancement of intelligent systems. The rapid growth of artificial intelligence and machine learning has seen robots and automated systems become integral across various industries [3-5].

The concept of learning from failure is not new [2, 6, 7], but it has gained renewed interest in the context of reinforcement learning. While Grollman and Billard's work on Gaussian models for learning from failed demonstrations laid the groundwork [7], recent studies have expanded on these ideas [8, 9]. For instance, approaches such as Hindsight Experience Replay have been developed to use failed episodes by reinterpreting goals, thereby improving learning efficiency [9]. Other methods focus on dynamically adjusting exploration strategies based on past failures to avoid redundant or harmful actions. By situating the Deep Q-Network-Donut Mixture Model (DQN-DMM) within this broader context, our work contributes to the growing body of research that seeks to enhance RL by systematically incorporating failure data, offering a robust alternative to traditional exploration methods.

By integrating these techniques into a variety of fields, we seek to enhance the generalizability and practical utility of learning from failure and exploratory trajectory generation. This expansion is essential for developing more versatile and adaptive AI systems capable of operating in diverse environments and contexts. We hypothesize that by incorporating these advanced learning techniques, we can significantly improve the efficiency and effectiveness of RL in real-world applications, where the stakes are often higher and the margin for error is minimal.

This paper will begin by discussing the theoretical underpinnings of the Donut Mixture Model (DMM), a novel approach that incorporates modified Gaussian distributions to penalize actions leading to known failures. We will then explore the integration of DMM into the Deep Q-Network (DQN) framework, providing detailed mathematical formulations and implementation strategies. Following this, we will present experimental results showing the effectiveness of DMM-enhanced DQN in a robotic arm simulation tasked with complex object manipulation. Finally, we will compare our approach with traditional RL methods, highlighting the advantages of learning from failure in reducing training time and improving performance in real-world scenarios.

By the end of this study, we aim to show that integrating DMM into RL algorithms offers a robust solution for enhancing learning efficiency, particularly in environments where trial-and-error methods are impractical. This work contributes to the broader field of AI by providing insights into how learning from failure can be generalized across different domains, paving the way for more adaptive and intelligent systems.

II. METHODOLOGY

A. Learning From Failure

We build upon the previous work of Billard and Grollman, specifically employing our novel version of the "Donut Mixture Model", a modified Gaussian Mixture Model [7, 10] as illustrated in Fig. 1. Our Donut Mixture Model is implemented

979-8-3315-3100-3/24/$31.00 ©2024 IEEE

as a strategic enhancement to the reinforcement learning process, designed to help an agent systematically avoid repeating past failures. Unlike traditional reinforcement learning approaches that primarily focus on maximizing rewards, our DMM introduces a mechanism that penalizes actions based on their proximity to previously recorded failures. This model derives its name from its unique ability to "carve out" low-probability regions around these failures in the action space, creating a safeguard reminiscent of a donut's hole [2].

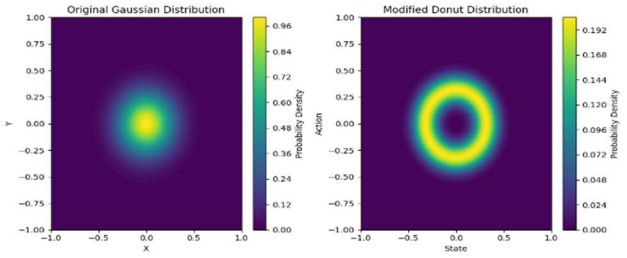

Fig. 1. Standard Gaussian distribution (left) and modified Donut Distribution (right). The Donut Distribution lowers the probability density at the center, representing a failure point, creating a "donut" shape that encourages the agent to avoid failure-prone areas.

Figs. 2 and 3 illustrate the impact of the Donut Mixture Model (DMM) on reinforcement learning. Fig. 2 shows how multiple failure points create "donut"-shaped penalty distributions, guiding the agent away from known failures. In Fig. 3, the left subplot shows the original reward function, while the right subplot demonstrates how the DMM modifies this reward by subtracting penalties near failure-prone regions, reshaping the reward landscape to steer the agent toward new and likely more rewarding actions.

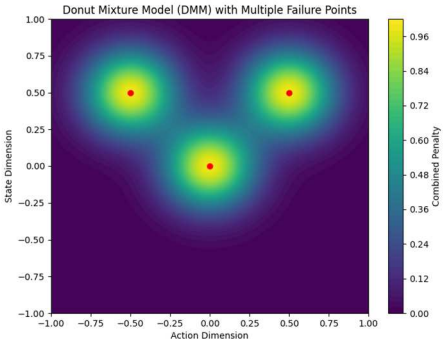

Fig. 2. The combined effect of multiple failure points. Each failure point contributes a "donut" distribution, and the overall penalty is the sum of these distributions.

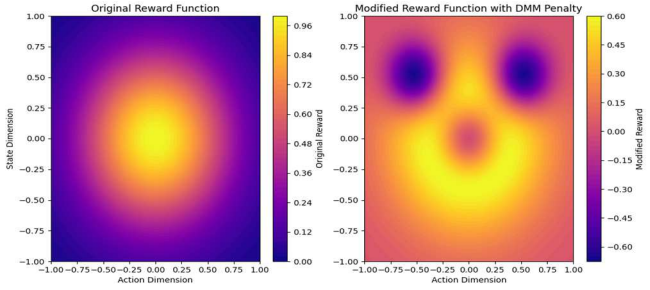

Fig. 3. Original reward and modified reward. The left image shows the original reward function, which might be high near a goal state or some optimal configuration. The right image shows how the DMM modifies this reward by subtracting penalties, creating a new reward landscape that steers the agent away from failure-prone regions.

At the core of our DMM is the Gaussian penalty function, which computes the penalty for a given state-action (s, a) based on its distance from known failure points (s_i, a_i). Each failure point contributes to the overall penalty, with closer failure points having a greater impact. The penalty $P(s, a)$ for a state-action pair is defined as:

$$P(s,a) = \sum_{i=1}^{N} \exp\left(-\frac{||s-s_i||^2+||a-a_i||^2}{2\sigma^2}\right), \quad (1)$$

where σ is the standard deviation of the Gaussian, N is the number of known failure points, and || || denotes the Euclidean distance.

The DMM can be integrated into the reinforcement learning framework by modifying the reward function. The standard reward is adjusted by subtracting the penalty from the DMM:

$$R'(s,a) = R(s,a) - \lambda \cdot P(s,a), \quad (2)$$

where λ is the scaling factor that controls the impact of the DMM penalty on the reward. In the context of a Deep Q-Network, the Q-value update rule is modified to account for the penalized reward:

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha\left[R'(s_t,a_t) + \gamma \max_{a'} Q\,(s_{t+1},a') - Q(s_t,a_t)\right], \quad (3)$$

where α is the learning rate, γ is the discount factor, and $\max_{a'} Q\,(s_{t+1}, a')$ represents the maximum Q-value for the next state. This update rule integrates the DMM penalty into the Q-value computation, steering the learning process away from failure-prone regions in the state-action space. This improved reward function ensures that the agent is discouraged from taking actions that lead to states near known failure points, effectively incorporating learning from failure into the reinforcement learning process.

B. DQN Implementation

We selected a Deep Q-Network for our project due to its advanced reinforcement learning capabilities, which are well-suited for handling the complexities of learning optimal policies in environments with high-dimensional state spaces. The Q-network is constructed as a deep neural network using the Keras framework, comprising three hidden layers. The network's input layer accepts the state vector representing the current environment. The network uses two hidden layers, with 512 and 256 nodes respectively, applying Rectified Linear Unit (ReLU) activation functions to introduce non-linearity and improve its ability to model complex functions. The output layer contains several nodes corresponding to the action space, each representing the Q-value associated with a specific action [11].

To stabilize the learning process, the agent employs an experience replay buffer, which stores transitions as tuples (state, action, reward, next state, done). During training, the agent randomly samples mini-batches of these transitions from the buffer. This random sampling helps break the correlation between consecutive transitions, reducing update variance and minimizing the risk of overfitting to recent experiences. The agent employs an epsilon-greedy policy for action selection [11]. Under this policy, the agent selects actions randomly with a probability epsilon, facilitating exploration of the action space. As training progresses, epsilon is gradually decreased, shifting

the agent's behavior from exploration to exploitation, where actions are chosen based on the highest predicted Q-values. This approach balances the need for exploring new strategies with the refinement of known successful actions, ultimately leading to the convergence of the learning process.

C. Training

The agent samples mini-batches from the experience replay buffer and updates the Q-network using the Bellman equation [12]. The loss function is computed as the mean squared error between the target Q-values and the predicted Q-values:

$$L(\theta) = E\left[\left(y_t - Q(s_t, a_t; \theta)\right)^2\right], \quad (4)$$

where $y_t = R'(s_t, a_t) + \gamma \max_{a'} Q\,(s_{t+1}, a'; \theta^-)$ is the target Q-value, with θ^- representing the parameters of the target network. $Q(s_t, a_t; \theta)$ is the predicted Q-value based on the current network parameter θ. The agent interacts with the environment over multiple episodes, collecting experiences and periodically updating the Q-network. The training loop continues until the agent reaches a predefined performance threshold or the maximum number of episodes is completed, which varies based on testing.

III. Results and Analysis

A. Cartpole Testing

We first implemented the DMM process in the CartPole-v1 environment from OpenAI Gym to evaluate our approach. The goal in this environment is to balance a pole on a moving cart by applying forces to keep the pole upright for as long as possible. The state space consists of the cart's position, velocity, pole angle, and angular velocity, while the action space allows the cart to be moved either left or right, as shown in Fig. 4. An episode ends when the pole falls beyond a certain angle or the cart goes out of bounds. The primary objective is to maximize the time the pole remains balanced. Our training loop continued either until the episode reward reached five hundred or the agent completed one thousand episodes.

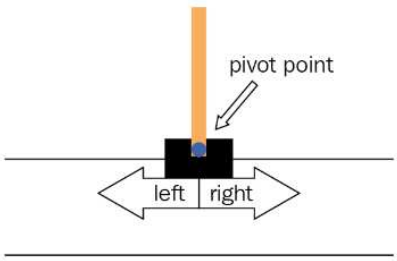

Fig. 4. Visual cartpole environment.

To evaluate the effectiveness of our approach, we conducted three tests using a standard DQN and three tests using our DQN-DMM method. Figs. 5a and 5b present the training-reward graphs comparing the performance of the DQN and DQN-DMM models. The DQN-DMM model demonstrated faster convergence, requiring approximately 14,000 steps to achieve optimal performance, compared to the baseline DQN, which required 20,000 steps. Moreover, the DQN-DMM model exhibited a more consistent increase in episode rewards, reflecting steady learning progress, whereas the baseline DQN displayed sporadic learning patterns.

Figs. 5c and 5d illustrate the testing performance over 10 episodes for the baseline DQN and the DQN-DMM models, respectively. The DQN-DMM model consistently achieved the maximum possible score of 500, indicating a highly effective learning process. In contrast, the baseline DQN managed a maximum reward of only 253, with significant variability in its performance. The results clearly demonstrate the superior and more efficient learning capability of the DQN-DMM model compared to the baseline DQN.

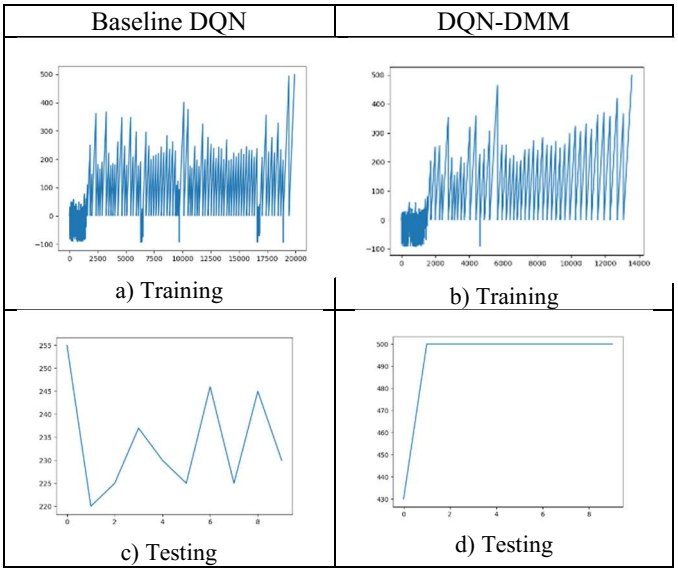

Fig. 5. Comparison of training and testing rewards for the CartPole environment using the baseline DQN and DQN-DMM models. a) Shows the training-reward graph of the baseline DQN model for approximately 20,000 steps. b) Shows the training-reward graph of the DQN-DMM model for approximately only 14000 steps. c) Shows the testing-reward graph of the baseline DQN model for 10 episodes in total, with a maximum reward of 223. d) Shows the testing-reward graph of the DQN-DMM model for 10 episodes in total, with it consistently scoring the maximum score available, 500.

B. Shower Testing

Following the success in the CartPole environment, we extended our testing to a custom environment: a shower simulation. The objective of this environment was to maintain the shower temperature between 37 and 39 degrees Celsius, which is considered the optimal range for showering. Each episode allowed for a maximum of 60 actions before the state reset. The state was defined solely by the current temperature of the shower, and the available actions were to increase the temperature, decrease the temperature, or do nothing. At the start of each episode, the starting temperature would be randomized, making the connections and the learning more difficult for the model to manage.

Like the CartPole environment, we conducted 3 tests using a DQN and 3 tests using our DQN-DMM method. Fig. 6 illustrates the reward graph comparing the performance of the DQN and DQN-DMM models. Remarkably, the DQN-DMM model outperformed the DQN model even more than the Cartpole test, exceeding our expectations. This further

demonstrates the effectiveness of the DQN-DMM approach in diverse and custom environments.

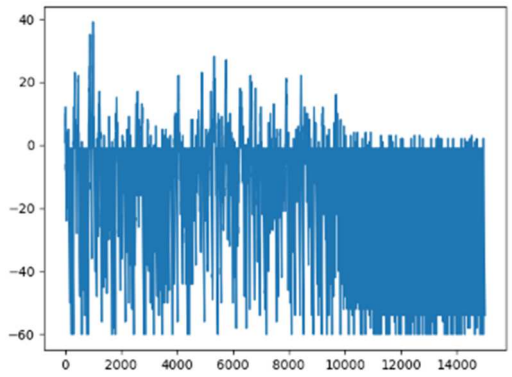

Fig. 6. The training-reward results of the baseline DQN model over 15000 steps, showing its struggle to achieve consistent improvements in performance.

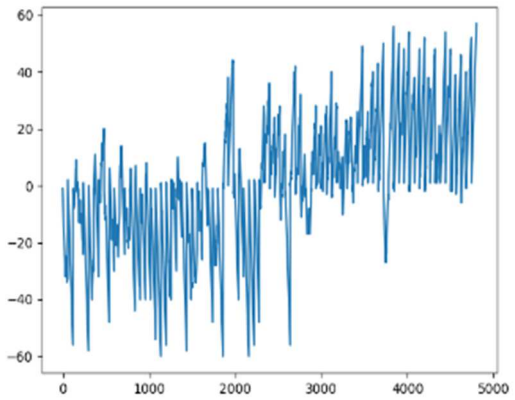

Fig. 7. The training-reward results of the DQN-DMM model after successfully completely training in only five thousand steps, demonstrating significantly faster and more stable learning compared to the baseline DQN.

Figs. 6 and 7 depict the training processes of the baseline DQN and DQN-DMM models, respectively. The baseline DQN struggled to develop a functional model even after 250 trials, which equates to around 14,000 steps, indicating significant challenges in learning and model formation. In stark contrast, the DQN-DMM model successfully trained a model in approximately 5,000 steps, demonstrating a remarkable improvement in learning speed—almost 300% faster than the baseline DQN. This substantial acceleration underscores the DQN-DMM's enhanced capability to establish connections and efficiently learn complex tasks, showcasing its clear superiority over traditional DQN methods.

C. Real-world Robot Application

Fig. 8 illustrates the setup of the robotic arm used in this experiment [13-16]. The objective was for the robotic arm to determine the correct combination of hole shape and angle for a square, using the force value, hole number, and degree of rotation as the state inputs. The environment provided 5 discrete actions for the robotic arm: rotate left by 2 degrees, rotate right by 2 degrees, switch to the square hole, switch to the circular hole, and switch to the triangular hole. Like previous tests, we conducted comparisons between 3 DQN models and 3 DQN-DMM models to evaluate the effectiveness of our approach in this new context.

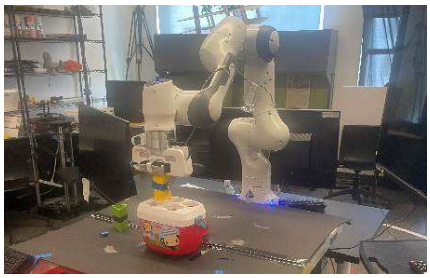

Fig. 8. Robot environment setup.

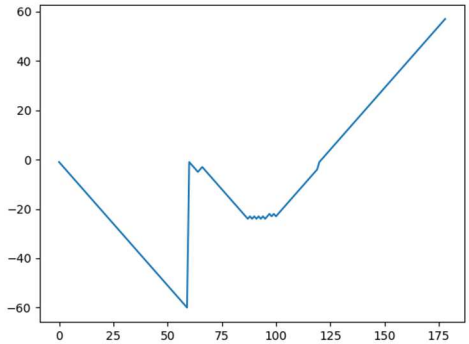

Fig. 9. The training-reward results for the DQN-DMM model over a span of 175 episodes.

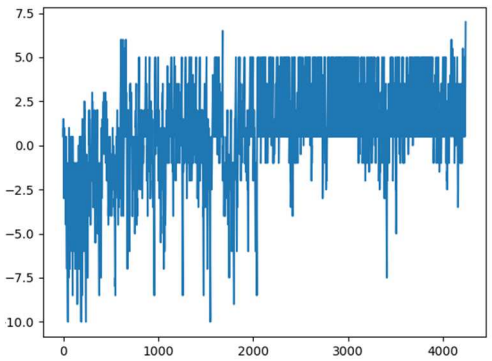

Fig. 10. The training-reward results for the baseline DQN model over a span of 4000 episodes.

Fig. 9 presents the training-reward graph for the DQN-DMM model, while Fig. 10 shows the same for the baseline DQN model. Consistent with the results from the CartPole and shower tests, the DQN-DMM model not only trained more quickly but also demonstrated a superior ability to establish correct connections throughout the training process.

D. Discussion

Figs. 5a and 5b illustrate that the DQN-DMM model reached optimal performance in approximately 14,000 steps, compared to 20,000 steps for the baseline DQN. This represents a 30% reduction in training time, highlighting the efficiency of the DQN-DMM. The DQN-DMM also demonstrated a steady increase in episode rewards, indicating a more stable learning process. In contrast, the baseline DQN showed sporadic learning, likely due to repeated exploration of failure-prone actions. The DQN-DMM's ability to penalize such actions effectively guided the agent toward more promising strategies, leading to faster and more consistent convergence.

In the testing phase (Figs. 5c and 5d), the DQN-DMM consistently achieved the maximum score of 500 across all episodes, showcasing its ability to generalize and perform reliably in varied scenarios. The baseline DQN, however, struggled with a maximum score of 253 and exhibited inconsistent behavior. This inconsistency suggests that the baseline DQN failed to adequately learn from failures during training, resulting in a less reliable policy.

The early training stages, as depicted in Figs. 6, 7, 9, and 10 further emphasize the DQN-DMM's efficiency. While for the shower test, the baseline DQN failed to develop a functional model after 14,000 steps, the DQN-DMM achieved this in just 5,000 steps—nearly 300% faster. For the real-world environment test, the baseline DQN achieved a maximum score of only 8 after more than 4000 episodes while the DQN-DMM model achieved a score of almost 60 in only 175 episodes. This rapid learning underscores the DQN-DMM's ability to effectively utilize failure data to accelerate the learning process, leading to quicker and more meaningful connections between actions and outcomes.

The DMM outperforms the baseline DQN primarily because it incorporates a structured approach to learning from failures. By penalizing actions near known failure points, the DMM enhances exploration, accelerates convergence, stabilizes the learning process, and results in the development of more robust policies. These improvements address the shortcomings of traditional DQN models, making the DMM a powerful tool for reinforcement learning, especially in complex environments.

IV. Conclusion and Future Work

To enhance the application of reinforcement learning in complex environments, we have developed and assessed a novel DQN-DMM model that integrates failure-aware mechanisms into the DQN framework. Our experiments, conducted across a range of environments from standard simulations like CartPole to custom scenarios such as shower temperature control, consistently suggest the DQN-DMM's superior performance in terms of training efficiency and robustness. The model's ability to effectively learn from failures positions it as a promising solution for real-world applications where traditional RL methods face limitations. The DQN-DMM model achieves higher accuracy and efficiency, effectively learning and adapting to diverse tasks. We have conducted real-world validation experiments that highlighted the model's ability to optimize decision-making processes in dynamic environments.

One area for future research is to further enhance the scalability and applicability of the DQN-DMM model in more challenging and varied environments. While the current model shows significant promise, future work should focus on streamlining computational efficiency and reducing training time for real-time applications. Overall, the DQN-DMM approach presents a robust and efficient solution for complex decision-making tasks, with ongoing research aimed at refining and expanding its capabilities for broader use cases.

Acknowledgment

This work is supported in part by the National Science Foundation under Grant CMMI-2138351 and in part by the National Science Foundation under Grant CNS-2117308.

References

[1] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research,* vol. 4, pp. 237-285, 1996.

[2] D. H. Grollman and A. Billard, "Donut as i do: Learning from failed demonstrations," in *2011 IEEE international conference on robotics and automation*, 2011: IEEE, pp. 3804-3809.

[3] W. Wang, R. Li, Y. Chen, Y. Sun, and Y. Jia, "Predicting Human Intentions in Human-Robot Hand-Over Tasks Through Multimodal Learning," *IEEE Transactions on Automation Science and Engineering,* vol. 19, no. 3, pp. 2339-2353, 2022, doi: 10.1109/TASE.2021.3074873.

[4] H. Diamantopoulos and W. Wang, "Accommodating and Assisting Human Partners in Human-Robot Collaborative Tasks through Emotion Understanding," in *2021 International Conference on Mechanical and Aerospace Engineering (ICMAE)*, 2021: IEEE, pp. 523-528.

[5] W. Wang, R. Li, Z. M. Diekel, Y. Chen, Z. Zhang, and Y. Jia, "Controlling Object Hand-Over in Human–Robot Collaboration Via Natural Wearable Sensing," *IEEE Transactions on Human-Machine Systems,* vol. 49, no. 1, pp. 59-71, 2019.

[6] K. Shiarlis, J. Messias, and S. Whiteson, "Inverse Reinforcement Learning from Failure," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 2016: International Foundation for Autonomous Agents and Multiagent Systems, pp. 1060-1068.

[7] D. H. Grollman and A. G. Billard, "Robot learning from failed demonstrations," *International Journal of Social Robotics,* vol. 4, pp. 331-342, 2012.

[8] T. Matsui, N. Inuzuka, and H. Seki, "Adapting to subsequent changes of environment by learning policy preconditions," *Int. Journal of Computer and Information Science,* vol. 3, no. 1, pp. 49-58, 2002.

[9] M. Andrychowicz *et al.*, "Hindsight experience replay," *Adv Neural Inf Process Syst,* vol. 30, 2017.

[10] H. G. Sung, *Gaussian mixture regression and classification*. Rice University, 2004.

[11] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602,* 2013.

[12] P. Dayan and G. E. Hinton, "Using expectation-maximization for reinforcement learning," *Neural Computation,* vol. 9, no. 2, pp. 271-278, 1997.

[13] E. Herrera, M. Lyons, J. Parron, R. Li, M. Zhu, and W. Wang, "Learning-Finding-Giving: A Natural Vision-Speech-based Approach for Robots to Assist Humans in Human-Robot Collaborative Manufacturing Contexts," in *2024 IEEE 4th International Conference on Human-Machine Systems (ICHMS)*, 2024: IEEE, pp. 1-6.

[14] J. Parron, T. T. Nguyen, and W. Wang, "Development of A Multimodal Trust Database in Human-Robot Collaborative Contexts," in *2023 IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2023: IEEE, pp. 0601-0605.

[15] O. Obidat, J. Parron, R. Li, J. Rodano, and W. Wang, "Development of a Teaching-Learning-Prediction-Collaboration Model for Human-Robot Collaborative Tasks," in *2023 IEEE 13th International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2023: IEEE, pp. 728-733.

[16] C. Hannum, R. Li, and W. Wang, "A Trust-Assist Framework for Human-Robot Co-Carry Tasks," *Robotics,* vol. 12, no. 2, pp. 1-19, 2023.