North American Academic Research

The TWASP Press

Original Research

Simulation Study on Weld Seam Path Planning of Battery Pack Welding Manipulator Based on ML-Agents

Pan Zijing[1], Yang Dapeng[1*],Shi Zhenyu[1],Sun Junma[1]

[1] School of Mechanical Engineering, Dalian Jiaotong University,794 Huanghe Road, Dalian,116028, China

Article history:
Received: **25** March 2026
Accepted: **28** March 2026
Published Online: **29** March 2026

***Correspondence:**
School of Mechanical Engineering, Dalian Jiaotong University,794 Huanghe Road, Dalian,116028, China

How to cite this article:
Pan Zijing, Yang Dapeng, Shi Zhenyu,Sun Junma (2026). Simulation Study on Weld Seam Path Planning of Battery Pack Welding Manipulator Based on ML-Agents . North American Academic Research, 9(3), 109-118. ***doi****: https://doi.org/10.5281/zenodo.19303035*

Abstract

Aiming at the pain points of long debugging cycle, insufficient precision and poor adaptability to 3D scenes in traditional path planning of welding manipulators for new energy vehicle battery packs, this paper builds a high-fidelity welding simulation environment based on Unity engine and ML-Agents toolkit, introduces the DQN deep reinforcement learning algorithm, designs a multi-dimensional composite reward function, and completes the training and verification of the weld seam path planning agent. Comparative experiments show that compared with the traditional Q-Learning algorithm, the DQN algorithm reduces the welding path error by 34.7% and improves the planning efficiency by 22%, which can effectively meet the high-precision production requirements of battery pack welding and provide a feasible solution for intelligent path planning of industrial welding.

Keywords: manipulator path planning; DQN algorithm; ML-Agents

Introduction

With the rapid development of the new energy vehicle industry, battery pack welding, as the core link of production, directly determines the safety performance of products. The precision and efficiency of weld seam path planning for welding manipulators further directly affect the welding quality. The traditional planning mode has obvious drawbacks: welding manipulators are costly, and path debugging deviations are likely to cause welding defects and result in battery pack scrapping. Therefore, before the manipulator is put into production and the path scheme is implemented, virtual simulation technology can be used to simulate the whole welding process with high fidelity, complete path algorithm verification, and process optimization in the virtual environment, so as to meet the high-precision production requirements of battery pack welding.

In the field of welding manipulator path planning and virtual simulation debugging, Li Jianhui et al[1]built a laser welding virtual simulation training and teaching system based on Unity3D engine, and realized visual simulation and virtual operation training of the welding process through high-fidelity scene modeling and development of full-process interactive logic. Luo Yu et al.[2]completed 3D modeling and motion simulation of robots on the Unity3D platform according to the operation requirements of vertical pipe horizontal welding robots, developed a virtual

reality system for welding robots, and verified the adaptability of the Unity3D engine in motion simulation of industrial welding robots. Yang Fuchun[3]systematically analyzed the industry pain points of traditional manual path planning according to the application requirements of intelligent manipulators in automobile welding scenarios, carried out applied research on path planning of welding manipulators, and clarified the core value of virtual simulation technology in the debugging of automobile welding production lines. Zhang Shanglin et al.[4]carried out special applied research on welding technology to meet the lightweight manufacturing requirements of new energy vehicle battery packs, and clarified the core impact of weld seam path planning accuracy on the structural strength, sealing performance, and safety performance of battery packs, providing clear industrial scenario demand support for the research of battery pack welding path planning.

In recent years, the research on intelligent path planning simulation of industrial robots based on Unity ML-Agents toolkit has gradually become an industry hotspot. Liu Qing[5]carried out research on the application of Agent technology in industrial production to meet the needs of industrial production process optimization, verified the core role of agent technology in improving quality and efficiency in industrial scenarios, and provided a theoretical reference for the application of agent technology in welding production. Liu Shuo et al.[6]optimized the iteration mechanism and exploration strategy of the algorithm through the Q-Learning algorithm framework, completed the simulation verification of the agent path planning algorithm, and effectively improved the convergence speed and stability of the reinforcement learning algorithm in path planning scenarios. Huang Zihang et al.[7]Focused on the path planning problem in complex dynamic environments, designed a state space and reward mechanism suitable for dynamic scenarios through the Q-Learning algorithm, and verified the feasibility of reinforcement learning in robot path planning through simulation experiments. Wen Junjun et al.[8]proposed an improved fused DQN algorithm to solve the problem of high-dimensional state space adaptation in robot path planning, optimized the deep neural network structure and experience replay mechanism, and verified the significant improvement of the algorithm in path planning accuracy and convergence efficiency through simulation experiments.

In summary, the traditional Q-Learning algorithm has problems such as difficult Q-value table storage, slow convergence speed, and low planning accuracy when dealing with high-dimensional state space weld seam path planning problems. Therefore, taking the automatic welding of manipulators as the core goal, this paper relies on SolidWorks and 3ds Max to complete simulation modeling optimization, builds a virtual welding scene through Unity, configures the ML-Agents framework with Python, and introduces the DQN algorithm to realize autonomous learning and intelligent decision-making of manipulator weld seam paths. From the perspective of the overall system architecture, the research system can be divided into three parts:3D model construction and optimization, virtual simulation environment construction, and agent path planning and training. Each module cooperates and links to jointly ensure the accuracy and efficiency of manipulator weld seam path planning, and the overall system is shown in Figure 1.

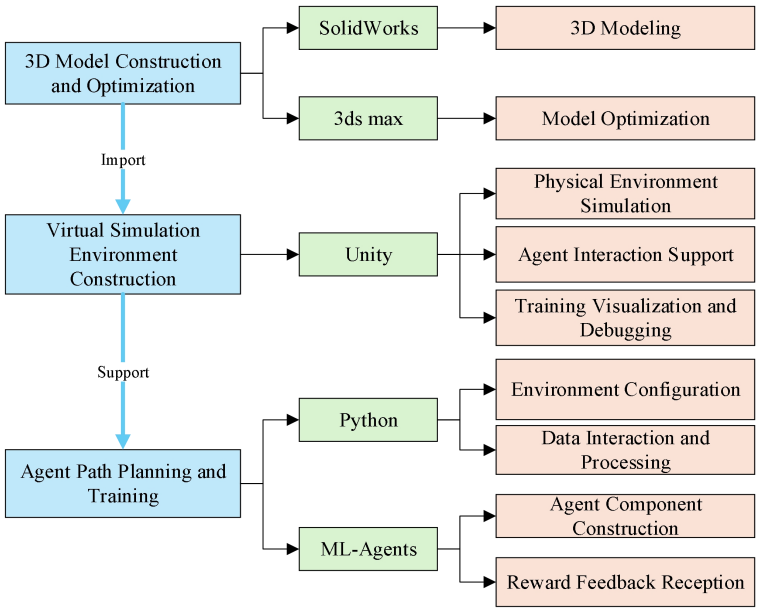

Fig. 1: Overall Design Framework of the System

2 Theoretical Framework

2.1 3D Model Construction and Optimization

The 3D model is the foundation of the simulation system, and its accuracy directly affects the simulation effect. The

mobile welding robot designed in this study includes an 8m-long slide rail and an IRB 5720-125/3.0 six-degree-of-freedom welding manipulator, with the manipulator parameters shown in Table 1. A battery pack for medium and large vehicles is adopted, with dimensions of 3200mm×1600mm×180mm and a weld seam width of 4mm.

Table 1: Manipulator Parameters

Joint Axis	Connecting Rod Length	Motion Range
Axis 1	670mm	+170°to-170°
Axis 2	320mm	+145°to-75°
Axis 3	1120mm	+70°to-180°
Axis 4	200mm	+300°to-300°
Axis 5	1547.5mm	+130°to-130°
Axis 6	167mm	+360°to-360°

Then, SolidWorks is used to conduct layered modeling of the welding manipulator by joints, and independent motion constraints are set for each joint to realize flexible simulation and precise control of the welding torch posture. Meanwhile, the 3D modeling of the battery pack shell is completed to clearly restore the cross-sectional shape of the weld seam and define the target trajectory for path planning, as shown in Figure 2. Finally, the model is imported into 3ds Max for simplification, including deleting redundant surfaces and merging vertices.

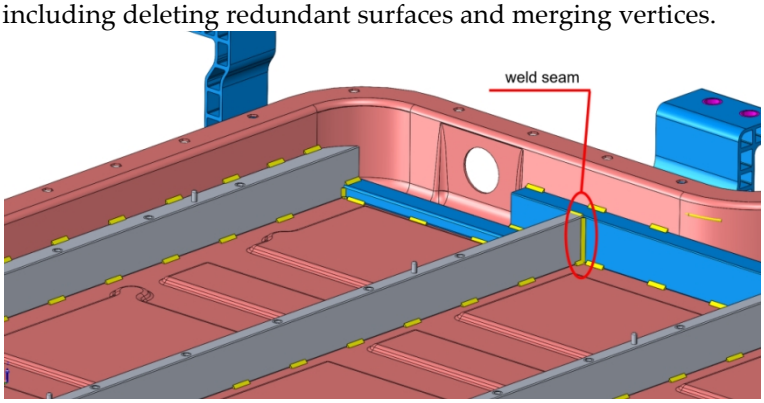

Fig. 2: Battery Pack Weld Seam

2.2 Implementation of IK Solution

Inverse Kinematics(IK)is the core algorithm for manipulator motion control. Contrary to the logic of forward kinematics(FK), which derives the end pose from joint angles, IK takes the target pose of the end effector as input and iteratively calculates the rotation parameters of each joint in the joint chain in reverse to achieve precise tracking of the target by the end. In the power battery pack welding simulation scenario of this study, the traditional FK control requires manual joint-by-joint debugging of rotation angles to match weld seam points, which is extremely inefficient and difficult to ensure accuracy. In contrast, IK can directly take weld seam feature points as the drive, automatically complete joint parameter calculation, and efficiently realize precise control of the welding torch, adapting to the high-precision requirements of welding operations.

In this study, Unity 2022.3.50f1c1 is used to build the simulation environment, and C#scripts are adopted to realize the core control logic, which is compatible with the ML-Agents plug-in to facilitate subsequent agent training and deployment. During the scene initialization stage, physical parameters are configured according to the real scene, the optimized model is imported, and the relative position between the welding torch and the weld seam is adjusted to a reasonable range. Firstly, a C#script is compiled to realize the IK solution, a joint array is defined, and a hierarchical relationship of the joint chain is established from the welding torch to the base in sequence. Then, the weld seam coordinate data set is imported, and the extracted feature points are set as IK targets with real-time pose updates. Finally, the angle of each joint is adjusted reversely according to the algorithm iteration, and the target pose is gradually approached through vector calculation and rotation solution. Simulation tests show that after importing three types of weld seam coordinate data and running the script, the manipulator can quickly respond to changes in the target pose, and the end positioning error is less than 0.01mm, meeting the accuracy requirements of battery pack welding.

2.3 Core Theories of Reinforcement Learning and ML-Agents

On the basis of realizing the precise motion control of the manipulator through the IK algorithm, the core of realizing intelligent autonomous path planning is to introduce a reinforcement learning framework to enable the manipulator to

independently learn the optimal weld seam tracking strategy through continuous interaction with the environment. The core principle of reinforcement learning is that the agent continuously adjusts its action selection strategy according to the reward signal fed back by the environment during the interaction with the environment, and finally converges to the optimal strategy that can maximize the cumulative reward. Its core elements include agent, environment, state, action, and reward, and the core interaction cycle is shown in Figure 3.

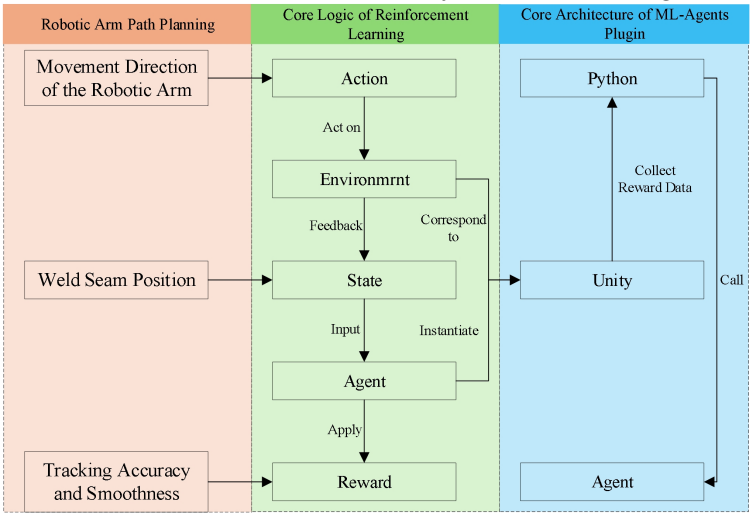

Fig. 3: Core Theories of Reinforcement Learning

The cycle consists of five core links: the agent perceives the current state of the environment, selects and executes an action according to its current strategy, the environment transitions to a new state after the action is executed, and feeds back an immediate reward to the agent according to the rationality of the action. The agent then optimizes its strategy based on the received state and reward signals, and enters the next cycle of interaction. In this study, the agent is the welding manipulator, the environment is the virtual welding workshop built in Unity, the state is the pose of the manipulator and the relative position between the welding torch and the weld seam, the action is the joint angle adjustment of the manipulator, and the reward is the multi-dimensional composite value designed according to path accuracy and motion smoothness.

The Unity ML-Agents toolkit, developed based on TensorFlow, is an open-source reinforcement learning framework that deeply integrates with the Unity engine. It is divided into two core parts: the Unity environment side and the Python training side. The environment side is responsible for the real-time operation of the 3D scene, the perception of the agent's state, the execution of actions, and the calculation and feedback of reward values; the Python training side is responsible for the operation of the deep reinforcement learning algorithm, the iterative update of the neural network model, and the transmission of the optimized strategy to the Unity environment. The toolkit natively supports mainstream deep reinforcement learning algorithms including DQN, adapts to single-agent and multi-agent training scenarios, allows users to customize state space, action space and reward function according to task requirements, and fully meets the technical requirements of weld seam path planning for welding manipulators in this study.

3 Design of Reinforcement Learning Algorithm

3.1 DQN Algorithm

The traditional Q-Learning algorithm is a classic model-free reinforcement learning algorithm, whose core is to iteratively update the Q-value table through the Bellman equation to realize state-to-value learning. The update formula is as follows:

$$Q(s_t,a_t)=Q(s_t,a_t)+\alpha\left[R_{t+1}+\gamma max_a Q(s_{t+1},a)-Q(s_t,a_t)\right] \quad (1)$$

Where is the learning rate, γ is the discount factor, and represent the state and action, respectively.

This algorithm has certain decision-making ability in simple low-dimensional scenarios, but in the high-dimensional state space of battery pack welding path planning, it suffers from slow convergence speed and low planning accuracy, making it difficult to meet actual production requirements. To solve this problem, deep reinforcement learning methods combining deep learning have emerged. Among them, the DQN algorithm is a deep reinforcement learning algorithm improved based on Q-Learning. Its core is to introduce a deep neural network to replace the Q-value table of traditional Q-Learning, solving the problems of Q-value storage and calculation in high-dimensional state space, making it more suitable for the welding path planning scenario in this paper. The deep neural network is used to fit the Q function, the experience replay mechanism is adopted to break the correlation of samples, and the target network is used to improve training stability, guiding the agent to select the optimal action and gradually approach the optimal path planning strategy.

In the Deep Q-Network algorithm, the fitting and updating of Q-values are realized relying on the deep neural network. The agent continuously interacts with the environment, and stores the interaction experience, including the current state s , executed action a , obtained reward, and new state after transition, into the experience replay pool. After randomly sampling batch samples from the experience replay pool, the target Q-value is calculated with the help of the target network, and the parameters of the current network are iteratively updated through the gradient descent algorithm. To ensure the stability of the training process, the parameters of the current network are regularly synchronized with the target network. The calculation expression of the target Q-value is as follows:

$$TargetQ = r + \gamma max_a Q(s', a^{\cdot}, \theta^{-}) \quad (2)$$

Where is the immediate reward, calculated based on factors such as the distance between the end effector and the manually extracted weld seam, and path smoothness? γ is the discount factor, representing the weight of future rewards. $Q(s', a^{\cdot}, \theta^{-})$ is the maximum Q-value corresponding to all actions in the next state output by the target network, and is the target network parameter.

Similar to other machine learning algorithms, DQN updates neural network parameters by minimizing the loss function. Based on the Bellman equation and the core iteration formula of Q-learning, the loss function of DQN is expressed as follows:

$$L(\theta) = E_{(s,a,r,s')\sim U(D)}[(r + \gamma max_a Q(s', a^{\cdot}, \theta^{-}) - Q(s, a^{\cdot}, \theta))^2] \quad (3)$$

Where U(D) represents uniform random sampling of experience samples from the experience replay pool D.The loss function of DQN is a residual model, which calculates the square of the difference between the true value and the estimated value. $Q(s, a^{\cdot}, \theta)$ represents the estimated value and also serves as the input of the neural network. The principle of the DQN algorithm model is shown in Figure 4:

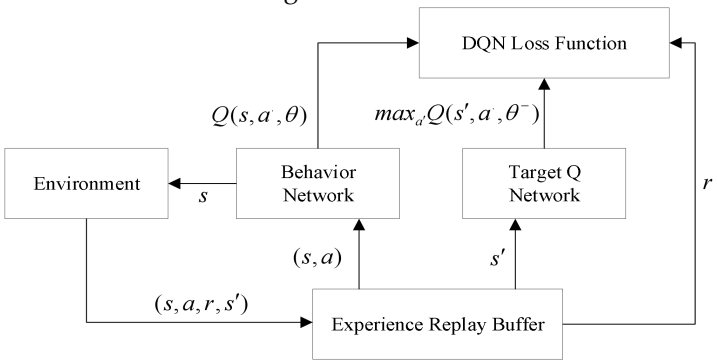

Fig. 4: Principle of the DQN Algorithm Model

The model consists of two core neural networks with the same structure, an experience replay pool, and a gradient update module. The evaluation network is responsible for selecting actions according to the current state of the agent and updating network parameters in real time. The target network is responsible for calculating the stable target Q value, and its parameters are periodically synchronized from the evaluation network. The experience replay pool stores the historical interaction experience of the agent, and random sampling is used to break the correlation of training samples. In this study, the input of the network is the state vector of the welding manipulator, and the output is the Q value corresponding to each optional joint adjustment action.

3.2 Reward Function Design

The reward function is the core of the DQN reinforcement learning algorithm, used to evaluate the pros and cons of the agent's actions and guide the agent to learn the optimal path planning strategy. Combined with the accuracy and efficiency requirements of battery pack welding, a multi-dimensional comprehensive reward function is designed to balance path accuracy and welding efficiency, ensuring that the reward signal can accurately feed back the rationality of the agent's actions and accelerate algorithm convergence. Based on this principle, the specific expression of the reward function R is as follows:

$$R = R_b + R_p + R_s - R_c - R_t \quad (4)$$

R_b is the basic reward, which motivates the agent to continuously perform welding path planning actions to ensure the orderly progress of the training process. It is a fixed value independent of the weld seam type. Considering training efficiency and reward signal stability, it $R_b = 0.5$ is set that the agent obtains a basic reward of 0.5 for each action interaction, regardless of the action's pros and cons, avoiding training stagnation caused by no reward feedback due to excessive initial action deviations of the agent.

R_p is the accuracy reward, which motivates the agent to approach the manually extracted weld seam coordinates and improve path accuracy, serving as the core of the reward function. Its value is positively correlated with the distance

between the end effector and the manually extracted weld seam feature points: the smaller the distance, the higher the accuracy reward. Let be the distance between the manipulator end effector and the current target weld seam feature point, calculated as $d = \sqrt{dx^2 + dy^2 + dz^2}$, where dx , and are the 3D position deviations between the end effector and the weld seam feature points defined in the state space.

The calculation formula for the accuracy reward of linear weld seams is as follows:

$$R_{p1} = 2.5 \times e^{-5d} \quad (5)$$

R_s is the smoothness reward, which motivates the agent to perform stable actions, avoids welding defects caused by manipulator motion jitter, improves path smoothness, and adapts to the welding requirements of complex weld seams of battery packs. The smoothness reward is calculated based on the variation of adjacent actions of the manipulator: the smaller the action variation, the higher the smoothness reward.

R_c is the collision penalty, which punishes the collision behavior between the manipulator and the battery pack, scene boundaries, etc, avoiding manipulator damage or welding position deviation and ensuring the safe and orderly welding process. Collision detection is realized through Unity's Collider component, and the collision penalty is triggered when any part of the manipulator collides with the battery pack or scene boundaries.

R_t is the timeout penalty, which punishes the agent for failing to complete the weld seam path planning within the specified time, improving path planning efficiency and meeting the high-efficiency requirements of industrial production.

The multi-dimensional composite reward function ensures the continuous progress of training through basic rewards, guides path accuracy with core accuracy rewards, optimizes motion stability with smoothness rewards, and avoids bad actions with various penalty items, forming a complete guidance mechanism. It can ensure the rapid convergence of the DQN algorithm and generate optimal paths that meet the accuracy and efficiency requirements of battery pack welding.

3.3 Hyperparameter Selection of the DQN Model

To ensure the reproducibility of the experiment and the stability of the algorithm training, the key hyperparameters of the DQN model are selected and determined through multiple pre-experiments, and the specific values are shown in Table 2. The selection basis of each core hyperparameter is as follows:

Learning rate (α): The learning rate determines the step size of the network parameter update in each iteration. If the learning rate is too large, the model will easily oscillate and fail to converge. If it is too small, the convergence speed of the model will be too slow. The learning rate is set to 0.00025, which can balance the convergence speed and training stability.

Discount factor (γ): The discount factor determines the weight of future rewards in the cumulative reward. For the continuous weld seam path planning task, a larger discount factor is needed to enable the agent to focus on the long-term path completion effect. The discount factor is set to 0.99, which is suitable for the continuous decision-making scenario of this study.

Experience replay buffer size: The size of the replay buffer determines the number of historical experience samples that the model can store. A larger buffer can improve the diversity of training samples, but will increase the memory occupation and computational cost.

Batch size: The batch size is the number of samples used for each network parameter update. The batch size is set to 64, which can balance the training efficiency and the stability of gradient update.

Target network update frequency: The update frequency of the target network determines how many steps the evaluation network is updated before synchronizing the parameters to the target network. The update frequency is set to 10,000 steps, which can effectively avoid the oscillation of the target Q value and improve the stability of training.

Exploration strategy: The $\varepsilon - greedy$ strategy is adopted to balance the exploration and exploitation of the agent. The initial ε is set to 1.0, which enables the agent to fully explore the action space in the initial training stage. The minimum ε is set to 0.05, which ensures that the agent still has a certain exploration ability in the later training stage.

Table 2: Key Hyperparameters of the DQN Model

Hyperparameter	Value
Learning rate(α)	0.00025
Discount factor(γ)	0.99
Experience replay buffer size	1,000,000

Hyperparameter	Value
Batch size	64
Target network update frequency	10,000 steps
Initial ε (exploration rate)	1.0
Minimum ε	0.05

4 ML-Agents and Python Configuration

4.1 ML-Agents Package Configuration in Unity

First, the ML-Agents package is imported locally from the disk through the package.json file in the Unity Package Manager, and the "ML-Agents" option is confirmed to appear in the Unity menu bar after the import is successful, as shown in Figure 5. Then, the Agent script is mounted on the GameObject of the manipulator base, named WeldingRobotAgent.cs, and the core logic including state space definition, action space definition, and the multi-dimensional composite reward function designed in Section 3.2 is compiled into the script to realize the real-time state perception, action execution and reward calculation of the agent. Finally, the scene is run and debugged step by step to ensure that the agent can normally execute the action according to the instruction, the reward value can be calculated and fed back in real time, and the episode can be terminated normally when the termination condition is triggered.

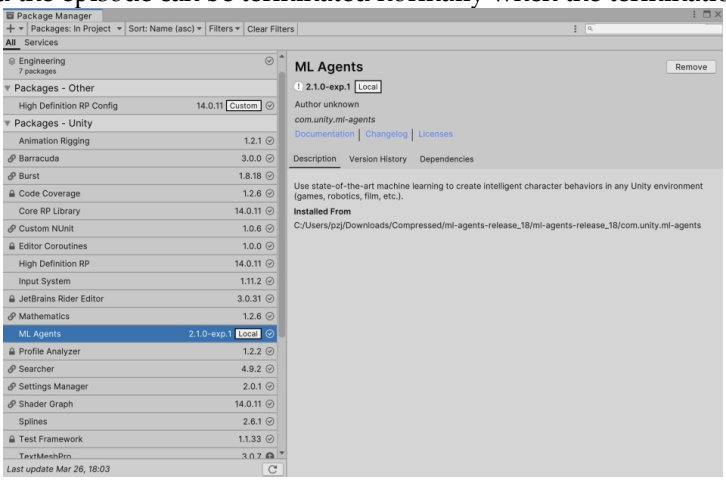

Fig. 5: Imported ML-Agents Package

The figure shows the successfully imported ML-Agents toolkit (version 2.1.0-exp.1) in the Unity Package Manager, including the core package com.unity.ml-agents and its dependent packages. The successful import of the toolkit is the basis for building the reinforcement learning agent in the Unity environment, and provides a bridge for the communication between the Unity simulation environment and the Python training end.

4.2 Python Environment Configuration

Next, we configure a Python virtual environment for algorithm training. Python 3.9.7 is selected for this experiment, which has good compatibility with the ML-Agents release_18 version used in this study. First, change the directory of the command window to the root path of the Unity project and create a dedicated Python virtual environment. Subsequently, activate the virtual environment. After entering the virtual environment, update pip to the latest version, and download and install the Python dependencies required by the ML-Agents toolkit via pip, including mlagents, tensorflow, pytorch, numpy, etc. Finally, run the Unity simulation scene and execute the mlagents-learn command in the Python virtual environment to start training. The successful communication interface between the Unity environment and the Python training side is shown in Figure 6, confirming that the environment configuration is completed and the agent can start formal training.

The interface shows that the ML-Agents toolkit has successfully established a two-way communication connection between the Unity simulation environment and the Python training end, and has identified the WeldingRobotAgent built in the scene. The successful communication means that the state data of the agent can be transmitted from the Unity end to the Python end in real time, and the action instructions and updated model parameters output by the algorithm can be transmitted from the Python end back to the Unity end, which is a prerequisite for the formal training of the reinforcement learning agent.

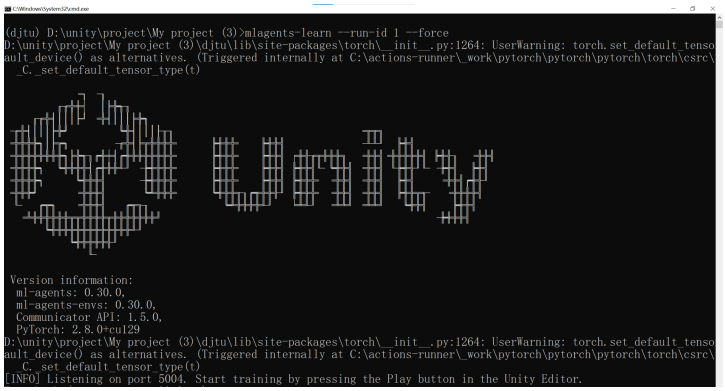

Fig. 6: Successful Communication Interface

5 Training and Testing

5.1 Construction of Virtual Training Environment

A high-fidelity industrial welding workshop scene is built in the Unity engine, as shown in Figure 7. The 6-DOF welding manipulator with the slide rail is arranged in the scene, and the joint control script and the WeldingRobotAgent script are mounted on the manipulator. A linear slide rail with a stroke of 8000mm is arranged and rigidly connected to the manipulator base to expand the horizontal operating range of the manipulator and adapt to the welding requirements of large-size battery packs. The target weld seams on the surface of the battery pack are clearly marked with colliders, and the environmental physical parameters, lighting effects, and environmental sound effects are configured according to the actual production workshop for subsequent linkage debugging and training.The scene is built with 1:1 scale according to the actual welding production workshop, and the physical properties and collision boundaries of all equipment are consistent with the actual scene, which can provide a realistic interaction environment for the training of the welding path planning agent, and ensure that the strategy learned by the agent in the virtual environment can be adapted to the actual production scene.

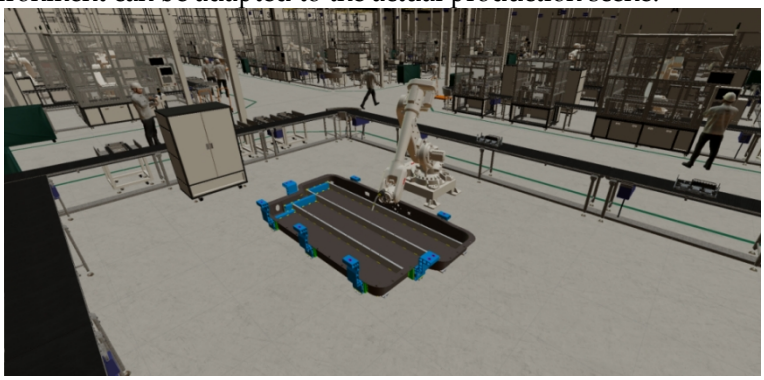
Fig. 7: Industrial Welding Workshop Scene

5.2 Training Process Design and Test Results

Training parameters are initialized, the experience pool is cleared, and the total number of training rounds is set to 10000. The process of each round is as follows: scene reset, state collection, action selection, action execution, reward calculation, experience storage, model training, target network update, and finally round termination judgment. The Unity scene model is constructed, and training is carried out in the Anaconda virtual environment. During training, reward value data is recorded in real time, a change curve is drawn, and training effects are monitored using the TensorBoard visualization tool to integrate training data.

From the experimental data fed back by TensorBoard, it can be seen that the cumulative reward value of the Q-Learning algorithm increases extremely slowly in the initial training stage, and the reward value tends to be stable only after 7000 training times, with a stable range of 30-50. The slow convergence speed and low stable reward value indicate that the strategy learned by the algorithm has poor performance and is difficult to adapt to high-dimensional weld seam path planning tasks. In contrast, the cumulative reward value of the DQN algorithm increases rapidly in the initial training stage, the agent clearly perceives the training intention after 3000 times, forms a mature behavior mode after 5000 times, and the reward value tends to be stable after 7000 times, with a stable range of 70-80. The convergence speed is increased by more than 50% compared with the Q-Learning algorithm, and the stable reward value is significantly higher, indicating that the DQN algorithm has better training stability and strategy learning ability. The change curves of the cumulative reward values of the two algorithms with the number of training times are

shown in Figure 8.

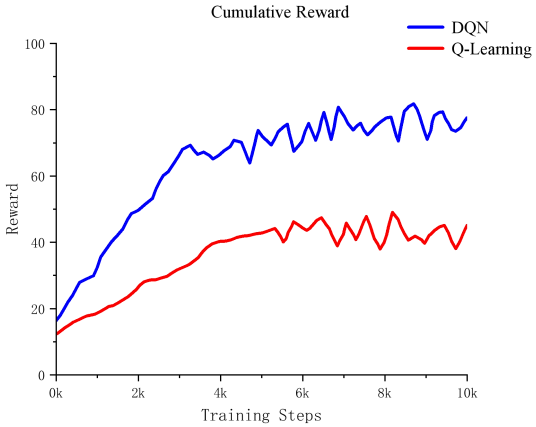

Fig. 8: Cumulative Reward Data Table of Training Models

After the training of the two algorithms is completed,100 successful planning rounds are selected for each algorithm, and the welding path error of the manipulator end effector and the average path planning time are counted. The results are shown in Table 3.

Table 3: Comparison of Algorithm Results

Planning Method	Average Welding Error(mm)	Average Planning Time(s)
Q-Learning Algorithm	0.95	100.4
DQN Algorithm	0.62	78.3

The average welding error of the DQN algorithm is 0.62mm, and the average planning time is 78.3s, representing a more than 34% reduction in welding error and a 22% improvement in planning efficiency compared with the Q-Learning algorithm(0.95mm average welding error,100.4s average planning time). Comparative experimental results show that the DQN algorithm used in this paper is significantly better than the traditional Q-Learning algorithm in three dimensions: convergence speed, path planning accuracy, and planning efficiency. It also greatly outperforms traditional manual planning, effectively solving the pain points of weld seam path planning for battery pack welding manipulators and adapting to the high-precision and high-efficiency requirements of industrial production.

In the last 100 successful rounds, the optimal solution is screened based on the evaluation criteria of minimum welding path error, shortest planning time, and no collision. The manipulator end coordinate sequence, joint motion parameters, and battery pack weld seam feature point coordinates of this round are imported into MATLAB. After kinematic verification and global welding sequence optimization, a complete 3D optimal path trajectory covering the battery pack weld seams is drawn, as shown in Figure 9, which intuitively verifies the global rationality and process adaptability of the path generated by the DQN algorithm.

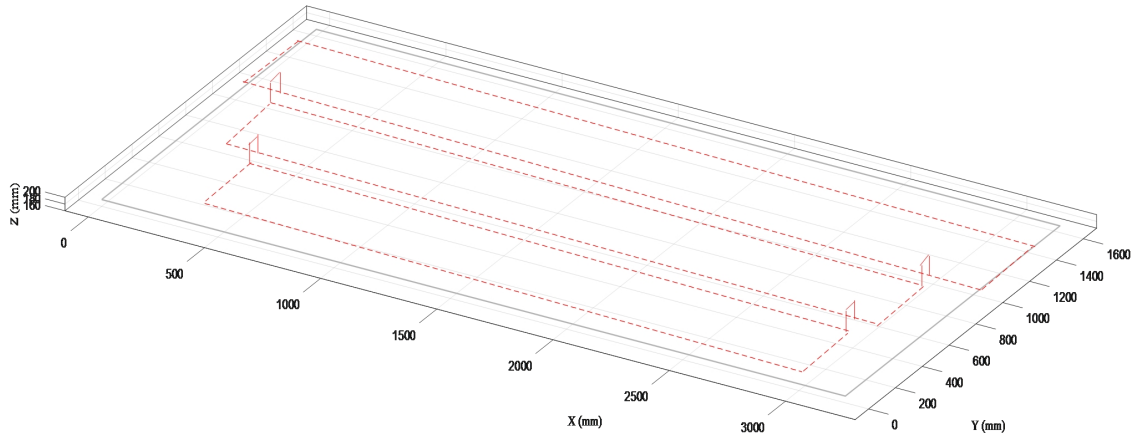

Fig. 9: Weld Seam Path Planning Trajectory of Battery Pack Welding Manipulator Based on DQN Algorithm

Conclusion

To address the core industry pain points of long debugging cycle, insufficient precision, and poor adaptability to high-dimensional 3D scenes in traditional path planning of welding manipulators for new energy vehicle power battery

packs, this paper conducts a simulation study on weld seam path planning based on the Unity ML-Agents toolkit and DQN deep reinforcement learning algorithm. This work completes 1:1 high-precision modeling and lightweight optimization of the IRB 5720 welding manipulator and target battery pack, builds a high-fidelity virtual welding environment, and realizes high-precision inverse kinematics solution for the 6-DOF manipulator. Meanwhile, we introduce the DQN algorithm, design a multi-dimensional composite reward function tailored for welding scenarios, and complete the training, convergence verification and performance test of the path planning agent. Comparative experiments show that, compared with the traditional Q-Learning algorithm, the proposed method reduces the average welding path error by 34.7% and improves the planning efficiency by 22%, breaking through the core bottleneck of traditional tabular reinforcement learning in high-dimensional state space. The research results provide a complete and feasible technical solution for intelligent path planning of industrial welding manipulators and pre-deployment virtual debugging of welding production lines, with important engineering application value for the intelligent upgrading of power battery pack welding production.

This study still has limitations in the adaptation of complex multi-type weld seam scenarios and the virtual-real migration of the agent strategy. Future research will be carried out from four core aspects to further improve the engineering practicability and scene adaptability of the proposed method. First, building a closed-loop digital twin system to realize high-fidelity virtual-real mapping and online iterative optimization of the path planning strategy. Second, introducing transfer learning to enhance the cross-scene reusability of the agent model for different battery packs and weld types. Third, integrating virtual PLC and industrial fieldbus to realize the seamless connection between simulation planning results and physical production lines. Fourth, fusing the DQN algorithm with other intelligent algorithms to improve the anti-interference ability of the method in complex dynamic production scenarios[9].

Author Contributions: At first page.
Approval: All authors have read and agreed to the published version of the manuscript.
Funding: This research received no external funding.
Institutional Review Board Statement: Not applicable.
Informed Consent Statement: Not applicable.
Data Availability Statement: Not applicable
Acknowledgments: Not Mentioned.
Conflicts of Interest: The authors declare no conflict of interest.

References

[1] Li J H, Yang B, Chen L, et al. Development of Laser Welding Virtual Simulation Training System Based on Unity3D. Internet of Things Technologies, 2025, 15(23):142-145.
[2] Luo Y, Ma Y, Guo J Z, et al. Research on Virtual Reality Technology of Vertical Pipe Horizontal Welding Robot Based on Unity3D. Machine Tool& Hydraulics, 2023, 51(12):171-175.
[3] Yang F C. Path Planning Research of Intelligent Manipulator in Automobile Welding. Auto Time, 2025, (24):126-128.
[4] Zhang S L, Wu S H, Qin Y, et al. Research on Application of Welding Technology in Lightweight Manufacturing of Battery Packs. Auto Time, 2025, (11):144-146.
[5] Liu Q. Application of Agent Technology in Industrial Production and Optimization of Process Flow. Modern Industrial Economy and Informatization, 2025, 15(12):140-141+144.
[6] Liu S, Dong X S, Zhao W.Research on Agent Path Planning Based on Improved Q-Learning Algorithm. Computer Era, 2025, (11):1-6.
[7] Huang Z.H., Yang C.Y., Ran Y.X. Research on Robot Path Planning Based on Q-Learning in Complex Dynamic Environments. Journal of Chongqing Technology and Business University, 2026, (10):1-9
[8] Wen J J, Li X. Robot Path Planning Based on Improved Fused DQN Algorithm. Computer Applications and Software, 2026, 15(11):1-10.
[9] Zhang X H, Tian C H, Lei M Y, et al. Path Planning of Drilling and Anchoring Robot Manipulator Based on Improved PPO Algorithm. Journal of China Coal Society, 2025, 50(12):5420-5433.


AN ISO 9001 : 2015 CERTIFIED JOURNAL
SCAN ME!
TO VIEW LATEST ARTICLES
@TWASP.INFO