IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Learning-Based Joint Control with Hierarchical Reinforcement Learning and On-Device Execution

Satoshi Yagi[1] and Jun Morimoto[1,2]

***Abstract*—In typical robot learning, deep reinforcement learning policies are employed in the upper control layer to generate target joint angles for robot motion, while conventional controllers are used in the fast lower control layer to control each joint motor. This paper presents a fully neural network-based hierarchical reinforcement learning approach for real-time robot joint control. The proposed method divides joint control into two layers: a high-frequency current control policy and a low-frequency position control policy. The current control policy drives the motor to follow the target current while learning the dynamic characteristics of the joint. The position control policy generates the target current to achieve a desired joint angle, allowing learning and inference at a slower frequency. By decoupling motor dynamics from position control, our method improves learning performance and enables policy generalization across joints. Experimental results on a three-joint robotic arm demonstrate the effectiveness of the proposed approach, including posture control using a shared position control policy across joints.**

***Index Terms*—Machine Learning for Robot Control, Embedded Systems for Robotics and Automation, Neural and Fuzzy Control**

I. Introduction

In typical robot learning systems, deep reinforcement learning policies are mainly employed in the upper control layer to generate target joint angles for robot motion, while conventional controllers, such as Proportional-Integral-Derivative (PID) controllers, are used in the lower control layer to control the robot's joints [1]. This structure allows the upper-layer learning module to focus on robot control at a slower frequency, such as positional information. While training at a slower frequency affords sufficient time for network inference and mitigates communication delays between the PC and the robot, it becomes challenging to capture environmental dynamics, such as force information, which change on a much faster time scale.

A major limitation of conventional lower motor control in robotic applications is that motor controllers require manual tuning when installed on a robot. For example, off-the-shelf motors come with preset PID gains, but when mounted on a robot, variations in load, friction, and mechanical structure often cause performance degradation. As a result, engineers need to manually adjust control parameters or rely on commercial auto-tuning software, increasing setup time and complexity.

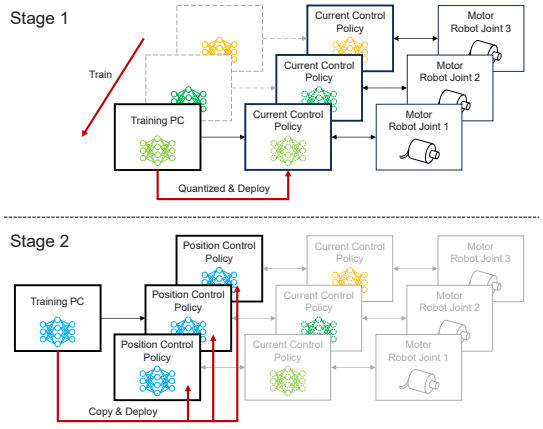


Fig. 1. The proposed method divides the joint control task into two stages: current control (Stage 1) and position control (Stage 2). Stage 1: The lower-layer network learns current control for each motor on a training PC. After training, it is quantized and deployed to a microcontroller on the robot. Stage 2: The upper-layer position control network learns to generate reference current values for the lower-layer current control policy. Since the lower-layer network manages joint-specific physical properties, the same position control policy can be used across all three joints.

The objective of this study is to enable neural networks to acquire both high- and low-frequency control of robot joints using deep reinforcement learning, eliminating the need for manual tuning or auto-tuning software. Our approach trains the entire control system through hierarchical reinforcement learning.

Fig. 1 shows our proposed hierarchical reinforcement learning framework, which divides the joint control task into two stages: position control and current control. By introducing a low-level current control layer, we can enhance learning efficiency and generalization. Specifically, the current control policy simplifies the learning problem by decoupling high-frequency torque control from the slower position control policy. This not only improves the sample efficiency of position control learning but also allows the learned policy to generalize across different joints and link configurations without requiring extensive retraining.

As shown in Fig. 1(a), the lower-layer network learns current control, operating at a high frequency of 1000 Hz. It outputs Pulse Width Modulation (PWM) signals to the direct current (DC) motor to ensure it follows the target current. This design allows the lower layer to focus on capturing

Manuscript received: May 25, 2025; Revised: July 25, 2025; Accepted: October 6, 2025.

This paper was recommended for publication by Editor Clement Gosselin upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by JSPS KAKENHI (JP23K16972, JP23K24925, and JP22H04998) and JST PRESTO (Grant Number JPMJPR23IA).

[1]Learning Machines Group, Graduate School of Informatics, Kyoto University, Kyoto, Japan

[2]Dept. of Brain Robot Interface, Computational Neuroscience Labs, ATR, Kyoto, Japan

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

the dynamics of the motor and attached link, enabling better generalization across different joint configurations. Another key advantage is that current control learning requires minimal motor movement, making it computationally and practically efficient, with a short learning time.

With practical implementation in mind, reducing memory and computational resource usage while ensuring real-time performance is crucial. By quantizing and deploying the learned network, inference can be performed on a microcontroller that can be mounted on the robot.

As shown in Fig. 1(b), the upper-layer position control network learns to output a target current value for the lower current control policy. The training of this network leverages the knowledge of the lower-layer current control policy, leading to efficient learning. We show that the same position control policy can be applied to all three joints by simply copying the trained network.

The contributions of this paper are as follows:

- We introduce presetting-free hierarchical reinforcement learning for robot joint current and position control, enhancing the learning efficiency of position control and outperforming the non-hierarchical approach.
- We validate the proposed method through real-world experiments on a robotic arm, demonstrating effective posture control by using the same network parameters across all position control policies.

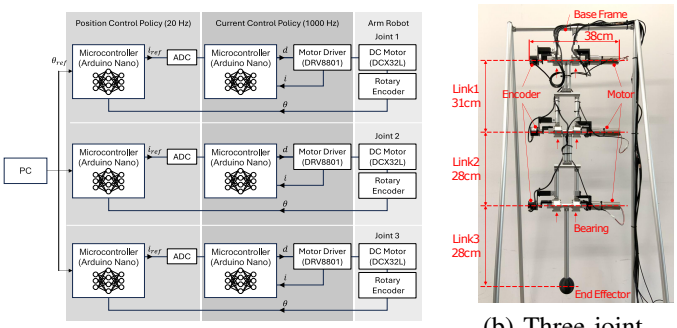
II. Related Work

The application of deep reinforcement learning (DRL) to motor control has been actively studied across various domains, ranging from standalone motor control [2], [3] to implementation in robotic systems [4]. One of the common approaches involves using DRL to determine the parameters of PID controllers for DC motor control [5]. For instance, Lu et al. [6] and Tüfenkçi et al. [7] applied DRL to design or compensate PID and PI controller gains. These methods adopt conventional PID control structures, making them easy to train and implement, as the learned policies do not need to run at high frequencies like in actual control scenarios.

On the other hand, studies aiming to acquire the entire controller through DRL also exist. For example, Mustafa et al. [8] applied Lyapunov-based DRL for speed control of ultrasonic motors. However, their control cycle operates at 50 Hz, leaving room for improvement in applications that require force control. Schenke et al. [9] proposed a DRL-based current control for PMSM motors, where the policy selects from a predefined set of discrete outputs. While effective, this method is limited by its discrete action space.

As an example of applying DRL to motor control on physical hardware, Poudel et al. [10] developed a steering control system for electric vehicles using DRL. Their method computes voltage control signals at 50 Hz on a PC and transmits them to a microcontroller. However, it does not sufficiently address high-frequency motor dynamics, making it less suitable for systems requiring fast control responses.

In robotic arm applications, Pajchrowski et al. [11] used DRL to learn a control policy that outputs a target current value. However, the final current control is still handled by a conventional controller. Kadokawa et al. [12] implemented a binarized convolutional neural network (CNN) on an FPGA to estimate the position of a ball from camera images, and used DRL to output discrete motor control commands based on that estimation. In both studies, motor-level control relies on predefined controllers, requiring manual parameter tuning when implemented on physical robots.



(a) System diagram (b) Three-joint robotic arm

Fig. 2. System configuration and the three-joint robot. (a) Each joint has hierarchical neural-based policies: one for position control and one for current control. (b) The robotic arm consists of three linkages. Each joint is equipped with a rotary encoder and a DC motor on both sides, with all joints oriented in the same direction.

Hierarchical reinforcement learning (HRL) provides temporal abstraction by decomposing tasks into options (subpolicies) [13], [14]. Its feasibility on real robotic systems has been demonstrated [15]. Our hierarchy operates at the actuator-control level, in contrast to prior work that modularly separates high-level goal planning from low-level motion primitives for robotic arm control [16]. Moreover, although hierarchical control with multiple time scales has been studied in optimal control [17], we realize this time-scale separation within a reinforcement learning framework by coupling a slower policy with a faster policy.

In this study, we apply HRL to robotic joint control by decomposing it into position control and current control. The current control policy, operating at a high frequency of 1000 Hz, is learned without relying on conventional motor controllers. Furthermore, by having the current control policy learn the joint-specific dynamics, the upper-layer position control policy can learn more efficiently in a simplified state space. This structure allows the position control policy to be reused across different joints, contributing to more generalizable and transferable learning.

III. Proposed Method

The proposed method applies hierarchical reinforcement learning to both position and current control in robotic joint control, as shown in Fig. 2. This approach allows the current control policy to focus on learning how to apply torque to the motor, while the position control policy concentrates on learning how to minimize the joint's angle error.

Fig. 2 shows the system overview (a) and the three-joint robot (b). In the system diagram (Fig. 2(a)), each of the three joints is equipped with two microcontrollers, one for position

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

control and the other for current control. The upper-layer position control policy operates at a frequency of 20 Hz and outputs a reference current value i_{ref} to track the target angle θ_{ref} provided by the PC. The reference current is transmitted as an analog voltage signal via a digital-to-analog converter (DAC) and is used as an input for the lower-layer current control policy. The current control policy operates at 1000 Hz, generating a PWM signal d to regulate the DC motor to follow i_{ref}. Based on these signals, the motor driver controls the DC motor. The position control policy receives feedback from a rotary encoder θ, while the current control policy receives current feedback i from the motor driver.

The robotic arm (Fig. 2(b)) consists of a base frame with three linkages, arranged from top to bottom as follows: Joint 1, Link 1 (1.4 kg), Joint 2, Link 2 (1.4 kg), Joint 3, and Link 3 (1.2 kg). Each joint is equipped with identical motors, which have a torque constant of 27.3 mNm/A and are coupled with a gearbox with a reduction ratio of 231:1.

For the robot, we used a DC motor (Maxon DCX32L), a motor driver (Pololu DRV8801), an optical encoder (Atyhao), a microcontroller (Arduino Nano 33 BLE), and a regulated 24 V power supply (Takasago ZX-S-400LA).

The training process follows a sequential approach: pre-training the current control policy, implementing it on a microcontroller through quantization, and pre-training the position control policy. Both policies are based on deep neural networks. We first complete this pipeline on a single motor, then fine-tune both policies and deploy them on six microcontrollers.

In this study, we evaluate two conditions: Proposed—the position control policy is fine-tuned once and shared across all joints, while the current control policy is fine-tuned independently for each joint; and Comparison—the current control policy is fine-tuned once and shared across all joints, while the position control policy is fine-tuned independently for each joint. Further details are provided in Section IV.

A. Current Control Policy

As shown in Fig. 2(a), the training is conducted on a desktop PC (Intel Core i9-13900KF CPU, NVIDIA RTX 4090 GPU). The PC communicates with a microcontroller for sensor control via serial communication, transmitting and receiving the target current value and measured current at a rate of 1000 Hz. During training, the motor is fixed in place using a jig to prevent movement of the output shaft, ensuring sufficient torque generation.

The current control network takes as input the current feedback of the DC motor and the target current value and outputs the PWM duty cycle to minimize the error. Let the target current be i_{ref}, the value measured by the motor driver's current sensor at time t be i_t, the error be $e_t^i = i_{\text{ref}} - i_t$. As the action a_t^i, the network outputs the PWM duty cycle d_t, which is then applied to the motor driver. The state is defined as:

$$s_t^i = [e_t^i, |e_t^i|, e_{t-1}^i, e_{t-2}^i, d_{t-1}, d_{t-2}] \tag{1}$$

To make the current control policy explicit, we denote all associated variables with a superscript i.

Since the current sensor readings from the motor driver contain significant noise, in this study, a low-pass filter with a cutoff frequency of 10 Hz is always applied to the measured values for training. The reward function is designed to mini-

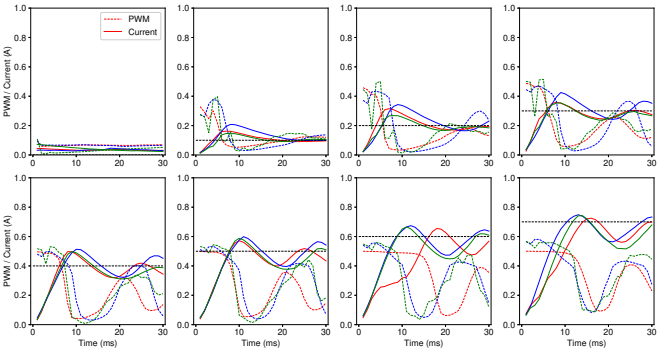


Fig. 3. Current-tracking performance for three motor types. Each plot corresponds to a target current ranging from 0.0 A (top left) to 0.7 A (bottom right). The dashed black lines indicate the target current. The dashed lines represent the PWM duty cycle output by the current control policy, while the solid lines show the measured current (low-pass filtered) from the motor driver. The horizontal axis represents time (ms), and the vertical axis represents the PWM duty cycle (-) and current (A).

mize the absolute error between the measured current and the target current:

$$r_t^i = -|e_t^i| \tag{2}$$

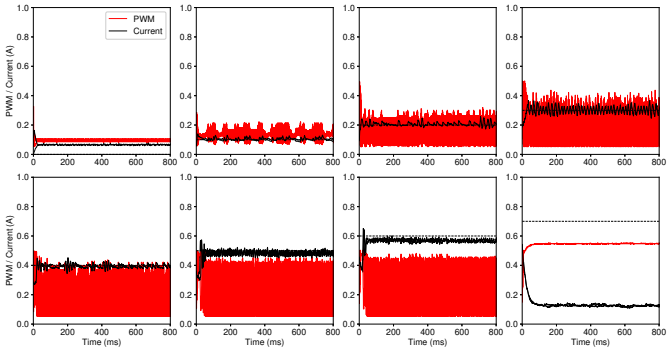
Additionally, a bonus reward of $\frac{0.1}{|e_t^i|}$ is given if the absolute error is less than 0.1. An additional reward of 100 is awarded if the absolute error remains below 0.05 for both the $t-1$ and t steps. To prevent overcurrent situations, a penalty of -50 is imposed when the measured current i_t exceeds the target current i_{ref}.

Proximal Policy Optimization [18] is employed as the reinforcement learning algorithm. We use a custom TensorFlow 2.0 implementation with discount factor 0.999, clipping parameter 0.30, batch size 32, two PPO epochs per update, and the Adam optimizer. Note that, we manually tuned the reward function and the PPO hyperparameters in this study. The neural network consists of two-layer Multi-Layer Perceptrons (MLPs) for both the Actor and Critic networks, with the number of nodes set to 8 in each layer. Batch normalization and Leaky ReLU (with 0.01) are applied to each layer. The Actor network outputs both the mean and standard deviation for a Gaussian distribution.

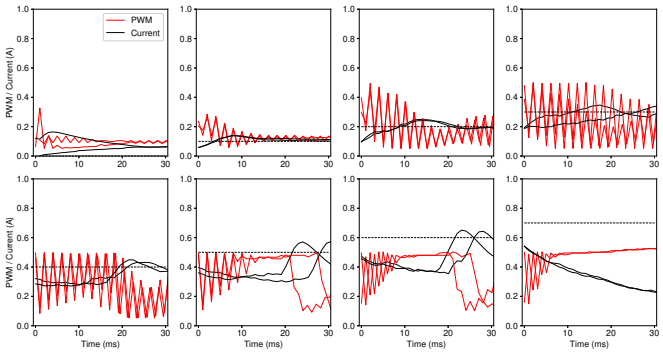
The learning rate α is set to 0.003. Training consists of 400 episodes, each with 30 steps of 1 ms, yielding an episode duration of 30 ms. The total training time was less than 7 minutes, including a 1-second environment reset for motor cooldown at the beginning of each episode.

The maximum target current value was determined through the following procedure. With the motor shaft fixed, the PWM duty cycle was gradually increased in increments of 0.1. At a duty cycle of 0.8, the maximum continuous current observed was 0.7 A. Beyond this point, the continuous current exceeded the motor driver's allowable limit of 1.0 A, triggering the protection circuit and resulting in almost no current flow. Based on these measurements, we set the target current range

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



(a) Full-time measurement



(b) Magnified view of the first 30 ms

Fig. 4. Time-series plots of the PWM duty cycle and measured current values using the quantized current control policy. (a) Measurements up to 800 ms. (b) Magnified view of the first 30 ms. Each plot corresponds to a target current ranging from 0.0 A (top left) to 0.7 A (bottom right). Each trial was conducted twice. The dashed black lines indicate the target current. The red lines represent the PWM duty cycle output by the current control policy, while the black lines show the measured current from the motor driver. The horizontal axis represents time (ms), and the vertical axis represents the PWM duty cycle (-) and current (A).

to 0–0.7 A in this study. A curriculum-based training approach was adopted, where the target current was randomly selected within the range $\left[0.2 - \frac{0.2k}{400}, 0.5 + \frac{0.2k}{400}\right]$ according to the current episode k in a total of 400 episodes.

Fig. 3 shows pre-quantization current-tracking performance for three motors. Using identical PPO hyperparameters, we trained a policy on each motor. For each motor, the plot shows measured current (low-pass filtered; solid line) and PWM duty cycle (dashed line) in red, blue, and green, respectively. The motors (in color order) have torque constants of 27.3, 22.9, and 22.9 mNm/A and gear ratios of 231:1, 35:1, and 1:1 (direct drive). These results indicate that the proposed method learns effective current tracking across heterogeneous motors.

B. Quantization of the Policy Network

To enable real-time execution of the trained policy on a microcontroller, the policy network trained on a PC is quantized. Quantization involves converting floating-point numbers (float32) used for network weights and activation outputs into integer values (int8). The network quantization was performed using TensorFlow Lite, a library provided by TensorFlow [19].

Fig. 4(a) shows two measurement results of the quantized current control policy, while (b) presents an enlarged view of the 0 to 30 ms interval. The graph plots the PWM duty cycle output by the current control policy and the current sensor readings from the motor driver for target current values ranging from 0 to 0.7 A. It can be observed that the target current is reached within 30 ms for values ranging from 0 to 0.6 A. However, for a target current of 0.7 A, the current value decreases over time. This is considered to be due to the activation of the motor driver's overcurrent protection circuit, which prevents further current flow.

C. Position Control Policy

In position control policy learning, the goal is to output the target current value that minimizes the error between the DC motor output shaft angle and the target angle. Let the target angle be denoted as θ_{ref}, the DC motor output shaft angle at time t as θ_t, and the error be $e_t^\theta = \theta_{\text{ref}} - \theta_t$. As the action a_t^θ, the network outputs the target current $i_{\text{ref},t}$. The state s_t^θ is defined as:

$$s_t^\theta = [e_t^\theta, |e_t^\theta|, \Delta\theta_t, |\Delta\theta_t|, i_{\text{ref},t-1}] \tag{3}$$

where $\Delta\theta_t = \theta_t - \theta_{t-1}$. θ is normalized from the range of -180 to 180 deg to the range of -1 to 1. To make the position control policy explicit, we denote all associated variables with a superscript θ.

The reward function is defined as follows:

$$r_t^\theta = \exp\left(-\frac{(e_t^\theta)^2}{2}\right) + 3\exp\left(-\frac{(e_t^\theta)^2}{0.01}\right) \tag{4}$$

where the first term provides a general reward for reducing error, and the second term adds a stronger reward when the error is small.

The target angle is randomly selected within the range of [-180, 180] deg. The network has the same structure as the current control policy but with three layers, 32 nodes, and ReLU as the activation function.

The training process consists of 100 episodes, each with 100 steps, where each step corresponds to a 50 ms interval. The total training time is approximately eight minutes and a half. Since the inference of the position control policy is slow, quantization is optional. In this study, the unquantized model is directly implemented on the microcontroller.

D. Comparison with Non-Hierarchical Control

We compare the proposed hierarchical current-position control with conventional non-hierarchical position control. The non-hierarchical method, which directly controls PWM duty cycle d_t based on the target angle θ_{ref}, is trained using the same procedure as the aforementioned position control policy.

All position control training was conducted with the third joint of the robotic arm used in later experiments, but with Link 3 removed to eliminate the effect of gravity.

Fig. 5 shows the comparison result. The upper graph shows the time-series angle of the joint tracking the target angle. As indicated by the black dashed line, the target angle was sequentially set to 0, 30, -30, 60, and -60 deg at 5-second intervals. The red line represents the proposed hierarchical method, while the blue line represents the non-hierarchical

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

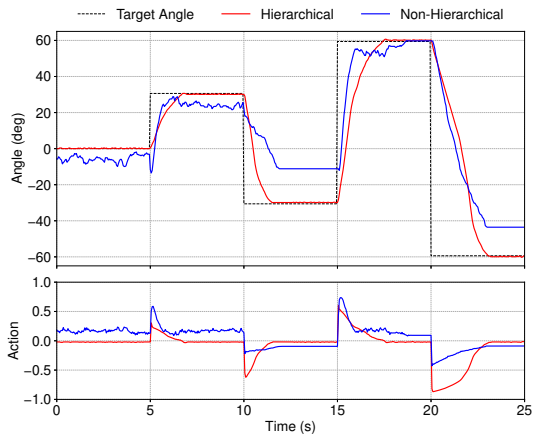


Fig. 5. Control performance of the non-hierarchical method and the proposed hierarchical method. The upper graph depicts the time-series joint angle tracking. The black dashed line represents the target angle, while the red and blue lines correspond to the angles achieved by the proposed hierarchical method and the non-hierarchical method, respectively. The lower graph depicts the action of the position control policy. The proposed method outputs the target current value (A), whereas the non-hierarchical method outputs the PWM duty cycle (-).

method. The lower graph shows the action of each position control policy.

From these results, it can be observed that the proposed hierarchical position-current control achieves better tracking performance than the non-hierarchical position control. Note that, in the hierarchical method, additional learning is required for the current control policy. However, as mentioned earlier, the actual motor operation time during the training of the current control policy is approximately 12 seconds, which is even shorter than the time required for three episodes of position control policy training.

IV. Experiment

To demonstrate the effectiveness of hierarchical robot joint control, we conducted three experiments on a robotic arm; responsiveness, waypoint-tracking performance, and robustness to disturbances. In the proposed method, the current control policy learns the dynamics of the joint and the connected link, allowing the position control policy to generalize across different joints. To evaluate this, we compare three conditions:

- **Shared position control policy condition**, where a single position control policy with shared network parameters is used across all joints, while individual current control policies are trained separately for each joint.
- **Shared current control policy condition**, where separate position control policies are trained for each joint, while a single current control policy with shared network parameters is used across all joints.
- **PID control condition**, where the PID parameters were tuned under a no-load setup using a single motor, and then applied to control the entire robot arm.

In the shared position control policy condition, the position control policy is fine-tuned at Joint 2 ($\alpha = 0.001$, 100 episodes) and then shared across all joints, while the current control policy is fine-tuned separately for each joint (α = 0.001, 100 episodes).

Conversely, in the shared current control policy condition, the current control policy is fine-tuned at Joint 2 (α = 0.001, 100 episodes) and then shared across all joints, while the position control policy is fine-tuned separately for each joint (α = 0.001, 100 episodes).

For the PID control condition, the PID parameters were initially determined using the same setup as in the previous section, where Joint 3 was used without Link 3. We adopted a cascaded position–current control scheme, comprising a position PID controller operating at 20 Hz (K_P = 8.5, K_I = 0.9, K_D = 0.085) and a current PI controller operating at 1000 Hz (K_P = 3.0, K_I = 1.0). The lower plot in Fig. 6(c) shows the performance of this configuration as the green line when the target angle was 50 degrees. These same PID parameters were then applied to all joints in the experiment.

A. Step Response Test

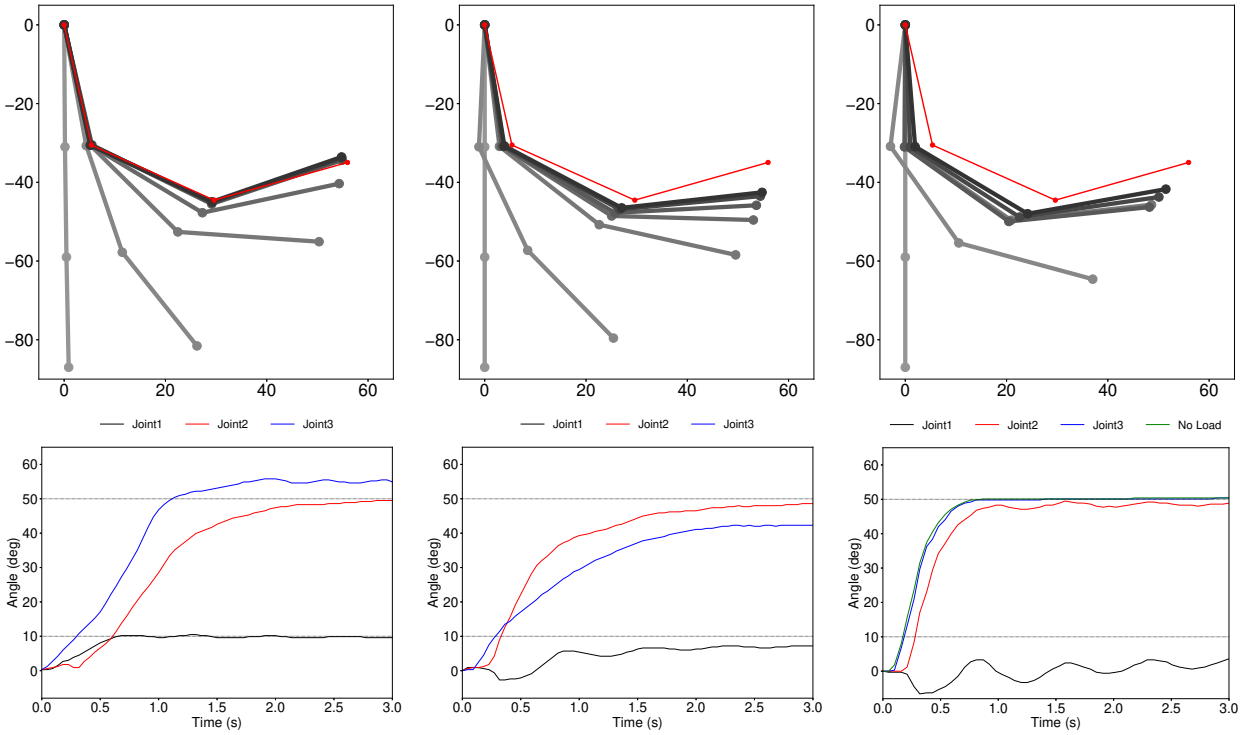
Fig. 6 shows the results of the step response experiment under three conditions. The upper part displays snapshots at 0.5-second intervals, with the red line indicating the target posture. The lower part shows the joint angles over time. The target angles for Joints 1, 2, and 3, indicated by the dotted black line, were set to 10, 50, and 50 deg, respectively. These values were selected within the range in which each joint can statically support the weight of its corresponding link.

At 3 seconds after the start of motion, the distances between the end-effector and its target position were 1.82 cm, 7.69 cm, and 8.13 cm for conditions (a) shared position control policy, (b) shared current control policy, and (c) PID control, respectively. Our method (a) outperformed the other two in terms of tracking accuracy.

In the shared position control policy condition, Joints 1 (black) and 2 (red) successfully reached their target angles. However, Joint 3 (blue) exhibited an average error of 4.89 deg after reaching the target. This was caused by the lightweight of Link 3, where the static friction of the motor's gearbox was relatively large, allowing the joint to maintain its posture due to friction when unloaded. As a result, the motor shaft did not rotate when the motor was not actively driven. Therefore, no current feedback was obtained, preventing the correction of the deviation. Fig. 7(a) shows frames extracted every 0.5 s, arranged left to right, under the shared position control policy condition.

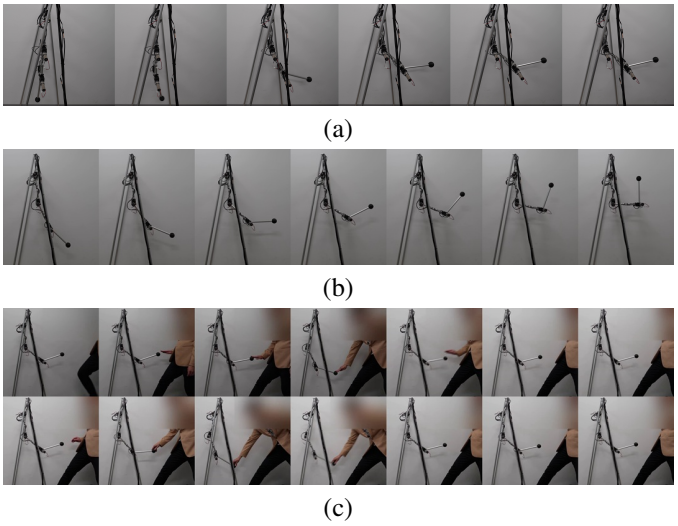
In contrast, under the shared current control policy condition, Joints 1 and 3 failed to reach their target angles, leaving noticeable deviations. Under the PID control condition, Joint 3, which bears a relatively light load, follows a trajectory similar to that observed when the motor is operated alone. In contrast, Joint 2, which supports a heavier load, exhibits a slight deviation from the target trajectory. Furthermore, Joint 1, which is responsible for lifting the entire arm, suffers from insufficient torque output, resulting in oscillations induced by the reactive motion of the other joints.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



(a) Shared position control policy (b) Shared current control policy (c) PID control

Fig. 6. Time-lapse visualization of the robot arm's reaching motion (above) and time-series plots of joint angles under three conditions: (a) shared position control policy, (b) shared current control policy, and (c) PID control. **Above**: Snapshots are shown at 0.5-second intervals from 0 to 3.0 seconds, with darker colors indicating later time steps. The red line represents the target posture. The plot is drawn with Joint 1 (the robot base) as the origin, with Joint 2, Joint 3, and the end-effector positions shown as circular markers extending downward. Both axes are in centimeters. **Below**: The solid black, red, and blue lines indicate the actual angles of Joints 1, 2, and 3, respectively, while the dashed black line denotes the target angle. The horizontal axis shows time (s), and the vertical axis shows joint angles (degrees). In (c), the green line additionally shows the position control trajectory when no load is attached to the motor.



(a)

(b)

(c)

Fig. 7. Sequential motion of the robotic arm under the shared position control policy: (a) step response test; (b) waypoint-tracking test; (c) disturbance-robustness test.

B. Waypoint-tracking Test

Fig. 8 shows the results of the waypoint-tracking test. Starting from an initial pose of approximately [0, 15, 15], the end-effector executes a waypoint-tracking motion in which the target is raised to the next (higher) black dot every 1.00 s, as shown in the figure. The three conditions are plotted as solid lines-red (shared position), blue (shared current), and green (PID). For each waypoint, the reached position is marked with a dot at the end-effector location that achieves the smallest position error relative to that waypoint. The motion was repeated three times per condition. Averaged across all trials and targets, the mean position errors were 2.84, 8.20, and 4.92 cm for the shared position (red), shared current (blue), and PID (green) conditions, respectively. These results show that our proposed method achieves better tracking performance across multiple waypoints. Fig. 7(b) shows frames extracted every 0.8 s, arranged left to right, under the shared position-control policy condition.

C. Disturbance-robustness Test

To evaluate the robustness of our proposed method, we conducted a disturbance test at the same posture as the step-response test, manually perturbing the end-effector. The supply voltage was also halved from 24 V to 12 V. Fig. 9 shows the end-effector displacement over time, and Fig. 7(c) shows snapshots of the test at 1.0 s interval. When a disturbance was applied at around 2–3 s and then released, the arm returned to approximately its nominal position within about 1 s and the oscillations decayed by 4 s. Repeating the disturbance at 8–9 s produced a similar recovery. The result indicates that the proposed method is robust to external disturbances.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

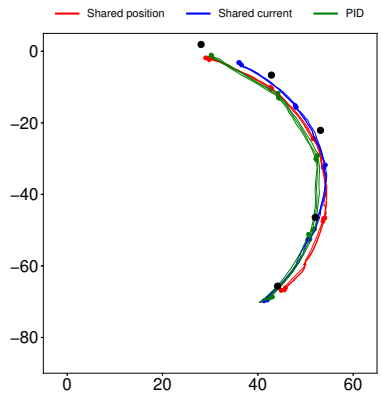


Fig. 8. Waypoint-tracking results. Three trials per condition are shown as solid lines—red (shared position), blue (shared current), and green (PID). Colored dots indicate reached positions and black dots indicate target positions. Both axes are in centimeters (cm).

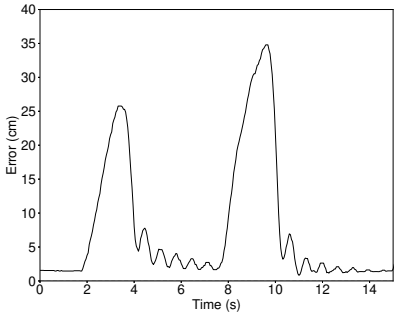


Fig. 9. Disturbance-robustness result: distance error from the target (cm) vs. time (s). External forces were applied at 2 s and 8 s.

V. Discussion

The proposed hierarchical reinforcement learning framework offers several practical advantages for real-world robotic systems. By introducing a learned current control policy, the method eliminates the need for manually tuned conventional motor controllers such as PID. This is particularly important when deploying the same robot in varying mechanical configurations or environmental conditions. Furthermore, the current control policy can be trained with minimal physical movement in very short episodes (milliseconds), reducing wear on the system and allowing fast, low-cost implementation.

The hierarchical structure also simplifies the state space of the upper-layer position control policy, making it more sample-efficient and enabling generalization across joints. Experimental results confirm that a shared position control policy can successfully control multiple joints with different physical dynamics, validating the effectiveness of this abstraction.

As an ablation study, we conducted the waypoint-tracking test with a configuration in which all policies were fine-tuned (combining the shared-position and shared-current settings). Across three trials, the average distance between the targets and reached positions was 9.56 cm. We attribute this degradation to a training–evaluation mismatch: during fine-tuning, the position policies for Joints 1 and 3 were optimized while running Joint 2's current policy, whereas at test time each joint used its own (different) current policy. This result indicates that, beyond fine-tuning the current policy, on-hardware fine-tuning of the position policy is also a key factor.

A. Limitations

While the hierarchical architecture improves control and learning efficiency, there are still several limitations to consider. First, although the current control policy captures low-level dynamics, it does not incorporate time-series memory explicitly (e.g., via recurrent networks), which may limit long-term error compensation—similar to lacking the integral term in classical PID control.

Second, one notable issue is the difficulty in learning actions that require large current outputs due to activation of the motor driver's overcurrent protection circuit. In our current implementation, the quantized current control policy failed to accurately follow the target current when the target value was set near the upper limit (e.g., 0.7 A). This limitation arises from the fact that overcurrent protection is triggered before such high-current actions can be successfully executed and reinforced.

B. Future Work

As a next step, the framework supports a mixed training strategy: train the position policy in simulation and the current policy on real hardware. In large systems, such as humanoids, this split reduces safety risks and data-collection costs.

To better cover a wider range of workloads, we plan to (i) train the position policy in simulation with payload and disturbance randomization; (ii) learn the current policy on hardware with minimal physical motion; and (iii) apply sim-to-real domain adaptation for the position policy, based on feature alignment or short real-world rollouts followed by imitation learning or RL [20], [21]. Another avenue is to add temporal models (e.g., LSTM/GRU) to the policy, which may reduce steady-state error.

VI. Conclusion

We proposed a hierarchical reinforcement learning method for robot joint control, combining high-frequency (1000 Hz) current control and low-frequency (20 Hz) position control. Experiments on a three-joint arm showed that sharing a single position control policy across joints is feasible with individually tuned current policies. The method outperformed non-hierarchical control, and quantized networks enabled real-time execution on microcontrollers. In the robot arm experiment, our approach achieved higher performance than both the shared current control policy condition and the conventional PID control condition.

References

[1] S. Lyu, X. Lang, H. Zhao, H. Zhang, P. Ding, and D. Wang, "RL2AC: Reinforcement learning-based rapid online adaptive control for legged robot robust locomotion," in *Robotics: Science and Systems*, 2024.

[2] A. Traue, G. Book, W. Kirchgässner, and O. Wallscheid, "Toward a reinforcement learning environment toolbox for intelligent electric motor control," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 3, pp. 919–928, 2020.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

[3] B. Cosmin and T. Sorin, "Reinforcement learning for a continuous DC motor controller," in *2023 15th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2023, pp. 1–4.

[4] Y. Ouyang, W. He, and X. Li, "Reinforcement learning control of a single-link flexible robotic manipulator," *IET Control Theory & Applications*, vol. 11, no. 9, pp. 1426–1433, 2017.

[5] K. Cheon, J. Kim, M. Hamadache, and D. Lee, "On replacing PID controller with deep learning controller for DC motor system," *Journal of Automation and Control Engineering*, vol. 3, no. 6, pp. 1–5, 2015.

[6] P. Lu, W. Huang, and J. Xiao, "Speed tracking of brushless DC motor based on deep reinforcement learning and PID," in *2021 7th International Conference on Condition Monitoring of Machinery in Non-Stationary Operations (CMMNO)*. IEEE, 2021, pp. 130–134.

[7] S. Tüfenkçi, G. Kavuran, and C. Yeroğlu, "An approach for DC motor speed control with off-policy reinforcement learning method," *Balkan Journal of Electrical and Computer Engineering*, vol. 11, no. 2, pp. 184–189, 2023.

[8] A. Mustafa, T. Sasamura, and T. Morita, "Robust speed control of ultrasonic motors based on deep reinforcement learning of a lyapunov function," *IEEE Access*, vol. 10, pp. 46 895–46 910, 2022.

[9] M. Schenke, B. Haucke-Korber, and O. Wallscheid, "Finite-set direct torque control via edge-computing-assisted safe reinforcement learning for a permanent-magnet synchronous motor," *IEEE Transactions on Power Electronics*, vol. 38, no. 11, pp. 13 741–13 756, 2023.

[10] B. Poudel, T. Watson, and W. Li, "Learning to control dc motor for micromobility in real time with reinforcement learning," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022, pp. 1248–1254.

[11] T. Pajchrowski, P. Siwek, and A. Wójcik, "Adaptive controller design for electric drive with variable parameters by reinforcement learning method," *Bulletin of the Polish Academy of Sciences. Technical Sciences*, vol. 68, no. 5, pp. 1019–1030, 2020.

[12] Y. Kadokawa, Y. Tsurumine, and T. Matsubara, "Binarized p-network: Deep reinforcement learning of robot control from raw images on FPGA," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8545–8552, 2021.

[13] R. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[14] T. Dietterich, "Hierarchical reinforcement learning with the maxq value function decomposition," *Journal of artificial intelligence research*, vol. 13, pp. 227–303, 2000.

[15] J. Morimoto and K. Doya, "Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning," *Robotics and Autonomous Systems*, vol. 36, no. 1, pp. 37–51, 2001.

[16] S. Takeda, S. Yamamori, S. Yagi, and J. Morimoto, "An empirical evaluation of a hierarchical reinforcement learning method towards modular robot control," *Artificial Life and Robotics*, pp. 1–7, 2025.

[17] K. Ishihara, T. D. Itoh, and J. Morimoto, "Full-body optimal control toward versatile and agile behaviors in a humanoid robot," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 119–126, 2020.

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

[19] R. David, J. Duke, A. Jain, V. Janapa Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, T. Wang *et al.*, "Tensorflow lite micro: Embedded machine learning for tinyml systems," in *Machine Learning and Systems*, vol. 3, 2021, pp. 800–811.

[20] X. B. Peng, E. Coumans, T. Zhang, T.-W. E. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," in *Robotics: Science and Systems*, 2020.

[21] S. Yamamori and J. Morimoto, "Foundational policy acquisition via multitask learning for motor skill generation," *IEEE Transactions on Cognitive and Developmental Systems*, pp. 1–11, 2025.