End-to-End Deep Reinforcement Learning for Exoskeleton Control

Lowell Rose, Michael C.F. Bazzocchi, *Member, IEEE*, Goldie Nejat, *Member, IEEE*

***Abstract*—Patient-specific control and training on lower body exoskeletons can help improve a user's gait during post-stroke rehabilitation by increasing their amount of participation and motor learning. Traditionally, adaptive control techniques have been used to provide personalization and synchronization with exoskeleton users, but they require predefined dynamics models of the user and exoskeleton. However, these models can be difficult to accurately define due to the complexity of the human-robot interaction. Most recently deep reinforcement learning techniques have shown potential to effectively learn control schemes without the need for system dynamics models. In this paper, we present for the first time an end-to-end model-free deep reinforcement learning method for an exoskeleton that can learn to follow a desired gait pattern, while considering a user's existing gait pattern and being robust to their perturbations and interactions. We demonstrate the effectiveness of our proposed method for user personalization of gait training in simulated experiments.**

Keywords—lower body exoskeletons, deep reinforcement learning, patient-specific control, exoskeleton control

I. Introduction

Lower body exoskeletons are increasingly being used for the gait rehabilitation of persons post-stroke [1], [2] and have been shown to be effective intervention tools for these patients [3]. They can aid in recovering gait symmetry, range of motion, increasing walking speed, as well as improving overall mobility and function [3]–[5].

Previous research on exoskeleton control has mainly focused on improving trajectory tracking, while imposing a specific gait pattern on a patient [6]–[8]. It has been found that while this provides some improvement in post-stroke recovery, it can also lead to a decrease in patient participation and motor learning, therefore resulting in less improvement in gait recovery for these individuals [6], [9]. Gait training that is specifically personalized for users has been found to be more effective at gait recovery [7], [10]–[13]. To achieve this, adaptive control techniques and learning-based control methods have been proposed; however, both often require a predefined dynamics model of the user and exoskeleton.

Adaptive control techniques use the dynamics models to represent the equations of motion of the exoskeleton, and determine the actuator torques necessary to control the exoskeleton in the presence of human interaction [9], [14]–[20]. To-date adaptive controllers allow exoskeletons to adjust and synchronize to users online [9], [14], [15], and try to account for some of the dynamic uncertainties and perturbations resulting from the human-robot interaction and variations in the behavior of different users [16]–[20]. However, due to the complex nature of the human-exoskeleton interaction, including the non-linear characteristics of this interaction and resulting dynamic uncertainties and perturbations, these models are not able to accurately and fully describe the interaction dynamics between the exoskeleton and user [17], [21]–[24].

More recently, reinforcement learning (RL) techniques have been used to improve upon the adaptive controllers by learning to optimize and personalize control parameters. RL allows for the automated iterative tuning of parameters by maximizing a reward [25], [26]. However, these RL techniques still must incorporate previously formulated dynamics models or must learn these models through transition probabilities of state-action pairs [27]. Furthermore, they can only learn in discrete observation spaces and produce discrete actions which are represented as control parameters (stiffness, damping) [26], [28]–[30] or discrete joint torques [31], due to the curse of dimensionality [32].

Model-free deep reinforcement learning (DRL) can potentially be used for exoskeleton control, where deep neural networks are used to represent the policy and value functions of RL. Their advantages over RL-based approaches are that they can learn high dimensional exoskeleton joint angles (which include position, velocity and acceleration information) and provide continuous actions such as joint torques, through only an agent interacting with a physics-based simulation environment [23]. Therefore, they do not require a previously defined or learned dynamics model. To-date, DRL has not yet been used in the control of exoskeletons in order to learn gait patterns.

In this paper, we investigate for the first time the development of an end-to-end DRL exoskeleton control method for user personalization of gait training. Our novel approach allows for model-free learning of a desired gait pattern, where the torque values of hip, knee, and ankle actuators of an exoskeleton are learned from scratch to achieve a desired gait, while considering a user's existing gait pattern and their perturbations. This unique control method can adapt the level of assistance needed as patients progress in their physiotherapy and therefore can be trained for individual patients.

This research is supported by the EMHSeed Fund, the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Canada Research Chairs Program. All authors are with the Autonomous Systems and Biomechatronics Laboratory of the Department of Mechanical and Industrial Engineering at the University of Toronto (lowell.rose@mail.utoronto.ca; michael.bazzocchi@utoronto.ca; nejat@mie.utoronto.ca).

II. Related Works

Previous work that has focused on the development of exoskeleton control with patient-specific adaptation can be categorized into: 1) adaptive control methods [18]–[20], [33]–[36], and 2) reinforcement learning methods [26], [28]–[31]. DRL techniques have been proposed for learning human and biped locomotion [22], [37]–[42], as well as learning upper and lower limb exoskeleton-based control policies [43]–[45], but have not yet been applied to end-to-end exoskeleton control for gait pattern training.

A. Adaptive Control Methods for Exoskeletons

Adaptive control techniques utilize dynamics models for both the user and exoskeleton in order to attempt to personalize to various users through adaptation of subject-specific parameters such as step length or walking speed [18], [34], or synchronize with a user's movements through online adaptation and feedback [19], [20], [35], [36].

In [18], a dual unscented Kalman filter was used to predict and generate exoskeleton trajectories based on the spatiotemporal features of a user's gait. This was accomplished with a partial coupled dynamics model of a human and exoskeleton, where the exoskeleton and user's limb were modeled as one body. An impedance supervisory controller was then used to follow a trajectory and synchronize with the user during their locomotion.

In [19], a compliance controller was used to provide motion control to an exoskeleton based on tracking a user's joint angles, foot pressure, and trunk inclination. A parameterized trajectory was created and deployed on the exoskeleton. The controller used the generated trajectory as equilibrium points, while allowing the exoskeleton to deviate slightly from these points when perturbed by the user in order to achieve compliance.

In [34], an exoskeleton was controlled using end point model predictive control (MPC) to create joint trajectories online. Step-length, swing duration and walking speed were used as inputs to create end point references for the MPC controller. This allowed for user-specific gait assistance within the swing portion of a gait pattern.

In [20], an assist-as-needed method was presented, where an exoskeleton only enacts force on a user when they cannot reach a desired goal on their own. An impedance-based force-field controller was designed to assist users by tracking the interaction forces between the user and the exoskeleton. The level of impedance was then adjusted based on whether the user was within the defined force-field. In [33], an impedance controller was also implemented to enable assist-as-needed control, by adjusting the end-point stiffness of the exoskeleton depending on the user's kinematic deviations in their gait phases.

In [36], a bipedal robot control design was adapted for an exoskeleton for spinal cord injury patients in order to allow them to walk without additional balance support. This method used a feedback controller, based on virtual constraints, to output a velocity regulating parameter comprised of the forward hip velocity of the exoskeleton, and posture regulating terms to synchronize the other joints. This method took advantage of offline trajectory optimization techniques to design gaits that could be tracked efficiently online.

In [35], a linear quadratic regulator (LQR) was used to improve trajectory tracking control on a lower limb exoskeleton, and to account for disturbances resulting from human-exoskeleton interactions. The LQR control inputs consisted of a feedforward reference torque, and feedback proportional, integral, and derivative actions. The parameters of the exoskeleton dynamics model were then derived experimentally through least squares estimation.

The aforementioned control methods require defined dynamics models of the exoskeleton and user. However, accurate models can be difficult to obtain due to the human-exoskeleton interaction and the resulting non-linear characteristics of this interaction [46], [47]. Therefore, typically, to facilitate the use of these adaptive control schemes, these models have been simplified by considering: 1) only the interaction forces between the exoskeleton and user [20], or 2) the human and exoskeleton as one body [18], [19], [34], [35].

B. Reinforcement Learning in Exoskeleton Control

Reinforcement learning (RL) has been used in exoskeleton control to: 1) optimize parameters of adaptive controllers [26], [28], 2) account for user-exoskeleton interactions [30], [31], or 3) provide assist-as-needed control [29].

In [26], dynamic movement primitives (DMPs) were used for an exoskeleton to model human movement trajectories and adapt to user motions. A coupled cooperative primitive (CCP) was defined, which incorporated an interaction term into a conventional DMP, using an impedance-based spring-damper model to describe the interaction. The CCPs were first learned through imitation learning using motion trajectories of users, then RL was used to update the stiffness, damping and scaling parameters of the CCPs. Using RL to learn these parameters resulted in the reduction of human and exoskeleton interactive forces and dynamic uncertainties.

In [28], RL was used for learning personalized parameters for an admittance controller, in order to reduce interaction forces between a user and a lower limb exoskeleton, while following a reference trajectory. The optimal parameters of stiffness and damping for the admittance controller were learned and tuned based on the performance of the user by using joint angle error and user-exoskeleton contact forces as observations. In [48], a similar method was employed for learning optimal impedance control parameters for an upper limb exoskeleton.

In [31], RL was used to design a model-reference compliance controller and to track a reference trajectory for a lower limb exoskeleton. Using Q-learning, the exoskeleton tracked a reference trajectory by observing discretized exoskeleton joint angles, velocities, and accelerations, and outputted joint torque values to a joint-based compliance controller. The compliance controller was added to incorporate safety features into the exoskeleton to allow for perturbations from the user, by modeling the exoskeleton joints as a second order mass-spring-damper system. The mass-spring-damper model was then used to account for interaction torque from the user and adjust joint torques accordingly.

In [30], assistive strategies for an upper limb exoskeleton were learned from interactions between an exoskeleton and user. Model-based RL was used to learn a dynamics model and control policies from limited data. The user moved a simulated arm with their muscle-activations recorded with electromyography (EMG), and a model-based policy search method found the optimal torque to assist the user in completing a reaching task and reduce the observed EMG signals.

In [29], an actor-critic based approach was used to model an impedance controller for assist-as-needed control of an ankle exoskeleton. The RL agent adjusted the level of assistance by altering the stiffness parameter of the force-field provided by the impedance controller, based on the user's performance. Performance was gauged by gathering observations of the tracking errors from a reference trajectory. This allowed for the controller to assist users in tracking a sinusoidal reference trajectory shown on a screen.

The aforementioned RL approaches were used for tuning or finding optimal parameters of a controller [26], [28]–[30] or for tracking a reference trajectory through modeled joints [31]. However, they also require a predefined dynamics model or need to learn the dynamics model by using the transition probabilities of state-action pairs. In general, RL is also constrained to discrete low-dimension observation and action spaces as it is not able to handle high dimensional or continuous systems [49]. However, exoskeleton control requires continuous high-dimensional observation and continuous action spaces to represent the joint angles, velocities, accelerations and actuator torques, and discretizing these spaces is difficult as there are too many state-action pairs to store [32].

C. *Deep Reinforcement Learning for Exoskeletons and Locomotion*

Deep reinforcement learning (DRL) has the ability to learn in the high-dimensional continuous observation and continuous action spaces by mapping the states to actions through the use of deep neural networks [32]. Therefore, it can overcome the limitations of standard reinforcement learning methods, while not requiring a predefined or learned dynamics model. The ability to learn in these spaces allows for more information to be provided to the controller, which can therefore output more accurate control torques [50]. To-date, model-free DRL has been used to learn: 1) human locomotion [37]–[39], 2) control of bipedal robots [22], [40]–[42], and 3) control of upper and lower limb exoskeleton joints [43]–[45].

In [37]–[39], DRL was used to learn locomotion skills for human musculoskeletal models. This was done with observations of joint angles, velocities, and accelerations, and joint torques or muscle activations as actions. DRL has also been successful in learning locomotion and motor skills for biped or legged robots, with stability, speed, and distance traversed as learning goals [22], [40]–[42].

With respect to exoskeletons, in [44], DRL was used to learn an optimal controller for enabling functional electrical stimulation (FES) of an upper limb while wearing a passive elbow exoskeleton. This was accomplished using a Proximal Policy Optimization (PPO) method. The learning agent took angular position, velocity and acceleration of the elbow, and the deviation from the expected goal as observations. This allowed for learning continuous actions of electrical stimulations to the arm muscles in order to enact elbow extension movements and for the user to reach specified elbow angles.

In [43], DRL was used to learn a fall predictor and recovery policy for an exoskeleton that was designed to assist in fall prevention of elderly users, by providing small torque adjustments to a user's locomotion when a potential fall was detected. PPO was used to learn a policy for human locomotion, and the simulated locomotion data was then used to train the fall recovery policy. This also was accomplished using PPO, with hip joint angular position and velocity as observations, and hip joint torques as actions. A support vector machine (SVM) classifier was trained to predict states that could lead to potential falling actions, and a hip exoskeleton was then added to validate the fall recovery policy and provided actuator torques to assist with maintaining stability.

In [45], a rehabilitation training game was presented wherein a character in the game was directed by a user wearing an EMG-controlled knee exoskeleton. The user's muscle activity was recorded from a thigh worn EMG sensor and translated into exoskeleton movements which controlled the game character. To enable the DRL-based assistance, a deep-Q network algorithm (DQN) was used to gather image-based observations from the game interpreted through a convolutional neural network (CNN), and produced outputs of discrete movement actions to control the character. To assist the user in playing the game, the DRL agent was able to augment the user's EMG-based control by converting the game movements into joint angles on the exoskeleton.

In general, the aforementioned DRL techniques have shown how DRL can be used to learn locomotion skills and help to control certain actions (elbow movements, maintaining balance). However, they have not focused on the specific problem of learning gait patterns for exoskeletons. End-to-end model-free DRL control methods, where actions are learned directly from raw sensor observations [51], can be specifically useful for such continuous and high-dimensional control tasks, as they do not require a dynamics model and can learn only from the exoskeleton directly interacting with the environment.

In this paper, we address the aforementioned limitations by developing a novel model-free DRL-based control technique that uniquely learns gait patterns for control of a lower limb exoskeleton. Our approach can be personalized for specific users by considering their specific gait patterns and perturbations, and observing their high-dimensional continuous hip, knee and ankle joint angles, velocities and accelerations, while providing appropriate actuator torques for the exoskeleton.

III. DRL FOR EXOSKELETON CONTROL

Our DRL approach utilizes the Deep Deterministic Policy Gradient (DDPG) [32] algorithm from the Keras-RL library [52] for end-to-end learning of a patient-specific torque controller for a lower body exoskeleton. Our overall architecture is presented in Fig. 1. The DDPG agent is composed of an Actor and a Critic neural network to represent the policy and value functions, respectively [32]. The actor network takes actions based on its current policy and observations from the environment, and the critic evaluates these actions based on the observations and a reward. The simulation environment is composed of a simulated exoskeleton model paired to a musculoskeletal model, implemented in OpenSim-RL [53]. In our model-free deep reinforcement learning approach, however, this model is not learned by the policy, with learning only conducted through the simulated model interacting with the environment. The details of our DRL approach are discussed below.

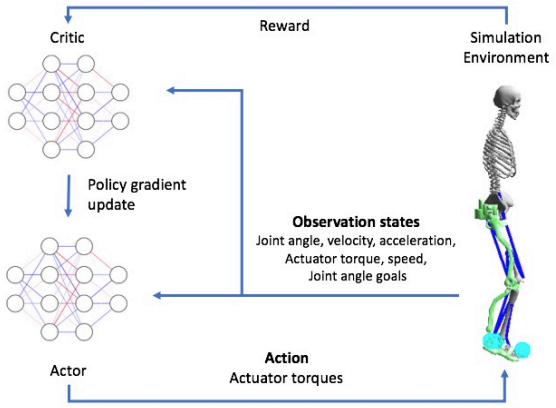

Fig. 1. DRL architecture overview for exoskeleton control.

A. Deep Deterministic Policy Gradient (DDPG)

DDPG is a model-free off-policy actor-critic deep reinforcement learning method [32]. The learning agent gathers an observation state from a fully-observable environment, s_t, takes an action, a_t, and receives a scalar reward, r_t, in each timestep, t, while maximizing the expected future discounted cumulative reward [32]:

$$R = \sum_{i=t}^{T} \gamma^{(i-t)}\, r_i(s_i, a_i)\,, \tag{1}$$

where $\gamma \in [0, 1]$ is a discounting factor.

The critic, $Q(s, a)$, is learned using the Bellman equation to represent the value function, which describes the expected return after taking actions. The actor policy, $\mu(s|\theta^{\mu})$, parameterized by mapping states to specific actions deterministically, is updated by following the policy gradient. Deep neural networks are used as function approximators [32].

During training, the critic network, $Q(s, a|\theta^{Q})$, and the actor network, $\mu(s|\theta^{\mu})$, are first randomly initialized with weights θ^{Q} and θ^{μ}. Target networks are also established, which are copies of the actor and critic networks, used to encourage a stable learning process and discourage divergence when updating the target value, Q [32]. These networks, Q' and μ', have the following weights:

$$\theta^{Q\prime} \leftarrow \theta^{Q} \tag{2}$$

$$\theta^{\mu\prime} \leftarrow \theta^{\mu}\,. \tag{3}$$

A replay buffer, B, used to store transitions of experience (s_t, a_t, r_t, s_{t+1}), is also initialized. While training, a random process $\mathcal{N}$ is initialized to encourage action exploration, based on an Ornstein Uhlenbeck Process [54], and a current observation state is received. The observations consist of the real-time joint angles for each hip, knee and ankle of the human model, the hip, knee and ankle actuator torque and velocity values of the exoskeleton, and the current goals. The goals at each timestep are comprised of the desired joint angles for the hip, knee, and ankle joints of the musculoskeletal model.

For each timestep, an action, $a_t = \mu(s_t|\theta^{\mu}) + \mathcal{N}_t$ is selected and executed based on the current actor policy, and a reward and new state are obtained. State transitions are then stored in the replay buffer. A target Q value is obtained from the target networks, and the critic is then updated by minimizing the loss between the critic network and target critic network, summed over a batch of state transition samples from the replay buffer. The actor policy is then updated based on the policy gradient [32]. The parameter optimizer for the actor and critic neural networks is based on the Adam optimizer [55], with a learning rate of 0.001 being used for the actor and critic networks.

The target actor and critic networks are updated based on the main actor and critic networks, but with a soft update so that they slowly track the main networks to encourage learning stability [32]:

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1 - \tau)\theta^{Q'} \tag{4}$$

$$\theta^{\mu\prime} \leftarrow \tau\theta^{\mu} + (1 - \tau)\theta^{\mu\prime}, \tag{5}$$

with $\tau \ll 1$.

As each timestep progresses, the desired joint angle goals are updated to the next joint angle in the gait patterns. Eventually the agent learns to reach all these joint angles in sequence in order to maximize its reward and deploy the desired gait pattern.

In our implementation of DDPG, the actor network is comprised of the state observations as inputs, three hidden layers with rectified linear unit (ReLU) activations, and an output layer of actions with a sigmoid activation function. The critic network takes actions and state observations as inputs, and is comprised of three hidden layers with ReLU activations and an output layer with a linear activation.

B. Reward Function

The reward function is comprised of the following components: 1) a reward based on the offset between the current and desired joint angles, and 2) a penalty for exceeding a maximum or minimum joint angle. This reward function allows the exoskeleton to follow a desired gait pattern by reducing the overall error between its joint angles and the desired joint angles, while providing additional penalties when an undesirable joint angle is reached. In this framework, a desired hip, knee and ankle joint angle is obtained from the desired gait pattern at each timestep, with the desired gait pattern updated at the beginning of each episode.

The total cumulative reward, R_E, for all joints, for each episode, E, over a number of steps, n, is defined as:

$$R_E = \sum_{i=0}^{n}(w_h r_{i_h} + w_k r_{i_k} + w_a r_{i_a})\,, \tag{6}$$

where r_i is the reward function for a particular joint, and h, k, a represent the hip, knee, and ankle joints, respectively. w is the weight for each joint.

The reward for each timestep r_i is defined as:

$$r_i = w_F \mathcal{F}(q_i) + w_G \mathcal{G}(q_i)\,, \tag{7}$$

where $\mathcal{F}$ and $\mathcal{G}$ are the components of the reward function, and w_F and w_G their weights. $\mathcal{F}$ represents a reward based on the current joint angle offset, and $\mathcal{G}$ represents a penalty for exceeding a maximum or minimum defined joint angle.

$\mathcal{F}(q_i)$ is defined as a Gaussian function:

$$\mathcal{F}(q_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{d-\mu}{\sigma}\right)^2}\,, \tag{8}$$

where μ and σ are the mean and standard deviation respectively, and d is defined as the absolute difference between the current joint angle, q_i, and the current desired joint angle, q_{d_i}:

$$d = \left|q_i - q_{d_i}\right|\,. \tag{9}$$

To assist in the learning process, a penalty is added if the defined maximum or minimum joint angles are exceeded during exploration, $G(q_i)$:

$$G(q_i) = -M(y_{\max}) - M(y_{\min}) , \tag{10}$$

where,

$$M(y) = \begin{cases} 0 & y \leq 0 \\ y & y > 0 \end{cases}, \tag{11}$$

$$y_{\max} = q_i - q_{\max} , \tag{12}$$

and,

$$y_{\min} = q_{\min} - q_i . \tag{13}$$

Above, $q_{\max}$ and $q_{\min}$ are the defined maximum and minimum joint angles, respectively, the values of which are set as 20 degrees beyond the maximum and minimum desired joint angles. With the use of a ramp function $M(y)$, the penalty will only be applied if the maximum or minimum joint angles are exceeded.

IV. User-Exoskeleton Simulation Environment

The simulation environment used for training the controller incorporates both a 3D musculoskeletal human model and a 3D exoskeleton model. OpenSim-RL [53] is used to provide the reinforcement learning environment and agent. This RL environment is based on OpenAI Gym [56], a popular platform for performing deep reinforcement learning experiments and benchmarking.

OpenSim-RL uses the OpenSim API platform to provide the physical simulation of the musculoskeletal model. This musculoskeletal model is derived from OpenSim's Gait2392 model, which is anatomically accurate and scaled to the proportions of a human with a height of 1.8 m and mass of 76.16 kg [57]. The OpenSim API platform has previously been used to perform dynamic simulations of locomotion or muscle control analyses [58]–[60]. The Simbody physics engine [61] is used to provide the multibody dynamics for the motion and interaction of the musculoskeletal bones, joints, muscles, and exoskeleton linkages in the OpenSim-RL environment. The simulation environment is presented in Fig. 2.

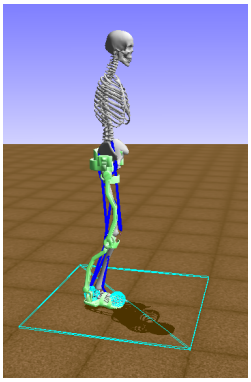

Fig. 2. Simulation environment in OpenSim-RL, composed of a musculoskeletal model with a hip-knee-ankle exoskeleton.

We adapt a 3D hip-knee-ankle exoskeleton model from [60] that contains a thigh, shank, and foot linkage, as well as a hip attachment. This is added to the OpenSim environment and secured to the musculoskeletal model, with torque actuators added to each joint (as would be found on a hip-knee-ankle exoskeleton). Each exoskeleton joint has a specified range of motion, in order to simulate a range similar to real exoskeleton joints [62]. The forces incorporated in the simulation environment consist of ground reaction forces on the bottom of the exoskeleton foot plates, acceleration due to gravity, and joint limit constraints. The exoskeleton parameters incorporated are summarized in Table I.

A hip-knee-ankle joint torque pattern, also obtained from the OpenSim Gait2392 model, is applied to the joints of the musculoskeletal model in order to simulate a post-stroke individual's baseline gait pattern. The corresponding joint angle pattern can be seen in Fig. 3. To obtain the desired gait pattern, Fig. 3, an inverse kinematic simulation is performed from motion tracking data provided by OpenSim [57] to generate a joint angle pattern for the hip, knee, and ankle.

TABLE I. Exoskeleton Parameters

Link	**Thigh**	**Shank**	**Foot**
Length [m]	0.47	0.42	0.26
Mass [kg]	1.5	1.5	0.5
I_{xx} [kg m^2]*	0.015	0.059	0.337
I_{yy} [kg m^2]	0.005	0.003	0.009
I_{zz} [kg m^2]	0.014	0.057	0.338
Joint Limits	**Hip**	**Knee**	**Ankle**
Maximum [°]	80	10	40
Minimum [°]	-80	-100	-40

*The I_{xx}, I_{yy}, and I_{zz}, are the moment of inertia tensor's diagonal values.

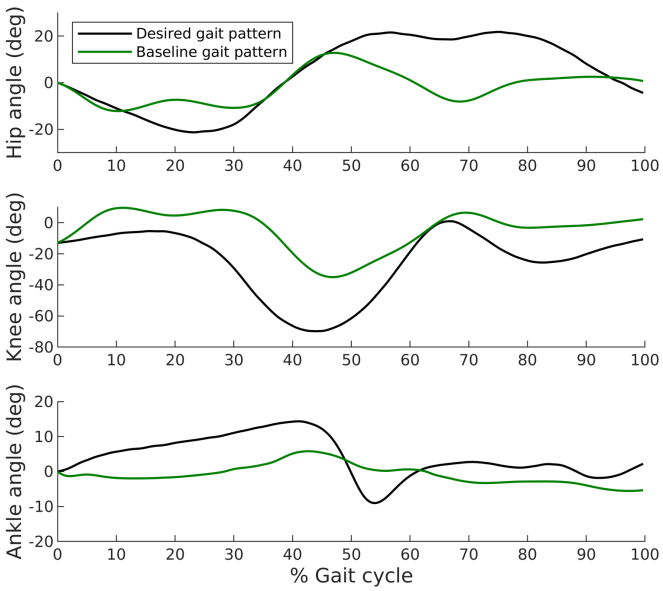

Fig. 3. Desired gait pattern and the user baseline gait pattern used in training.

V. EXPERIMENTS

In order to validate our DRL method for exoskeleton control of gait patterns, we first train the DDPG agent with the baseline and desired gait patterns. We then test the trained policy for controlling the exoskeleton joints.

A. Training

Training was conducted with the OpenSim-RL framework, with a user baseline gait pattern and a desired gait pattern as shown in Fig. 3. Training was implemented on an Intel Core i7-8700 CPU for over 2,200 episodes, with 308 timesteps per episode. After 1,300 episodes, the cumulative reward reached an average of approximately 63, as seen in Fig. 4. The parameters and weights for the reward function were defined as $\sigma = 5$, $\mu = 0$, $w_F = 1$, $w_G = 0.002$, and $w_h, w_k, w_a = 1$. The hyperparameters for the training and networks are as follows: $\gamma = 0.99$, $\tau = 0.001$, and a learning rate of 0.001.

B. Testing

Testing was conducted in two stages: 1) Stage 1, with the baseline and desired gait pattern used in training, and 2) Stage 2, with adjusted baseline and desired gait patterns from the training. Stage 2 testing was used to investigate robustness to slight changes in the gait patterns. Namely, the baseline joint torque and desired joint angle values for each joint were adjusted by a scale factor.

For both stages, the exoskeleton performed 10 full gait cycles of the learned joint torque pattern, with the resulting hip, knee, and ankle joint angles of the musculoskeletal model averaged over the cycles.

C. Results

The results of the tests in Stage 1 are presented in Fig. 5. The descriptive statistics for the error are also presented in Table II. It can be seen that the trained exoskeleton was able to follow the desired gait pattern closely throughout the gait cycle. The mean absolute error ranged from 1.06 degrees for the ankle to 2.63 degrees for the knee. The knee joint exhibited the largest error as it also has the largest range of motion of the three joints. The Stage 2 results for the scaled desired joint angle adjustments are presented in Fig. 6, with the descriptive statistics in Table III. Scaling factors of 0.5 and 0.8 applied to the desired joint angles were used to demonstrate deviations in the desired gait pattern that would be applicable during different stages of gait rehabilitation. It can be seen that the adjusted desired gait patterns were followed closely even though they were not used in training, with the mean absolute error ranging between 0.73 degrees for the ankle to 1.72 degrees for the knee. As the range of motion of the desired gait pattern in the Stage 2 tests were lower, the errors are also lower across all joints. In general, the error ranges from our results are comparable to existing model-based adaptive and RL-based joint control methods, which have ranged from 1 degree to 5 degrees, i.e., [35], [28].

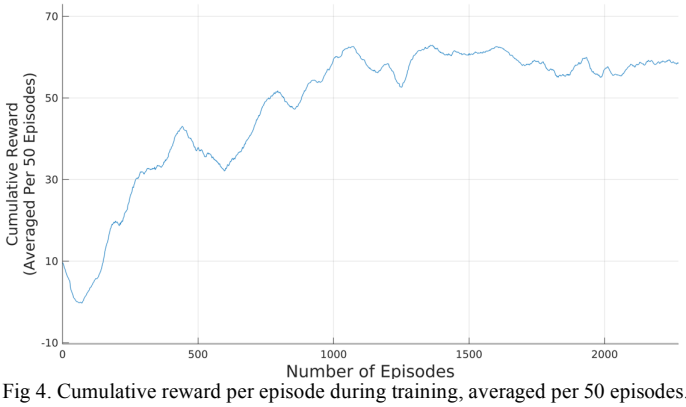

Fig 4. Cumulative reward per episode during training, averaged per 50 episodes.

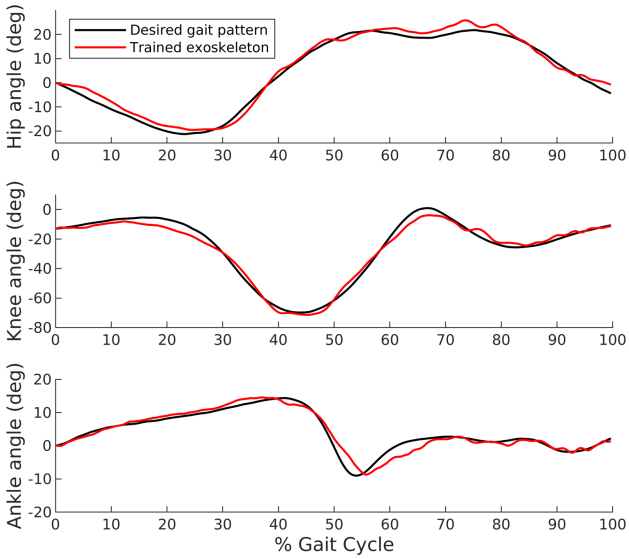

Fig. 5. Desired and the exoskeleton trained joint angle patterns for the hip, knee, and ankle joints.

TABLE II. STAGE 1 GAIT PATTERN ERROR

Exoskeleton joint	Mean absolute error (degrees)	Standard deviation (degrees)
Hip	1.82	1.01
Knee	2.63	1.72
Ankle	1.06	1.05

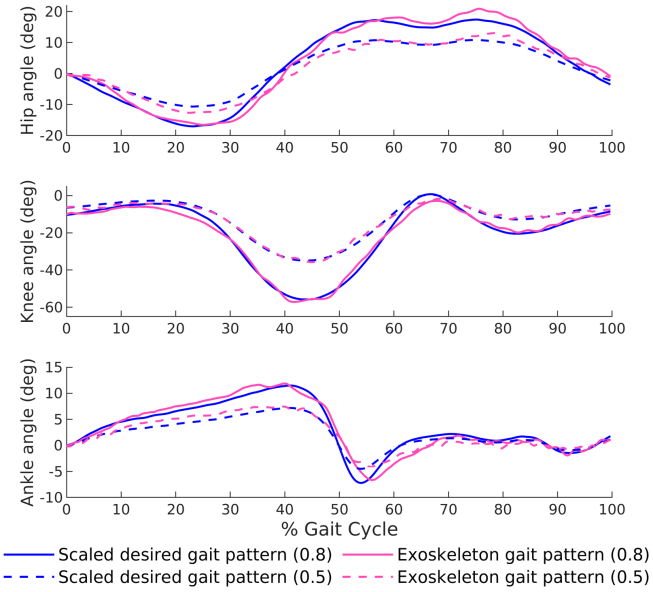

Fig. 6. Desired and exoskeleton gait patterns with the desired gait pattern scaled by a factor of 0.5 and 0.8.

TABLE III. STAGE 2 GAIT PATTERN ERROR

	Exoskeleton joint	Mean absolute error (degrees)	Standard deviation (degrees)
Scaled desired gait pattern (0.8)	Hip	1.58	0.98
	Knee	1.72	1.3
	Ankle	0.92	0.77
Scaled desired gait pattern (0.5)	Hip	1.5	0.92
	Knee	1.08	0.85
	Ankle	0.73	0.51

The results validate the effectiveness of our DRL method to learn and follow a desired gait pattern while accounting for a user baseline gait pattern as well as the ability to handle small deviations from the trained desired pattern.

VI. CONCLUSIONS

In this paper, a novel method for end-to-end DRL for exoskeleton control was developed. Our approach allows for user personalization of gait training using Deep Deterministic Policy Gradient (DDPG). Control torque values are learned for the exoskeleton hip, knee and ankle joints directly from observed joint information, without the need of a predefined or learned dynamics model. The DDPG agent was trained and tested in a 3D simulated physics environment. Experimental results showed that the learned torque control allowed the exoskeleton to closely follow the trained desired gait pattern as well as small deviations from it. The ability of our controller to personalize to users can help increase motor learning and function during post-stroke gait rehabilitation, leading to greater improvements in recovery. Our future work consists of evaluating the controller for use with additional desired gait patterns unseen in training.

ACKNOWLEDGMENTS

The authors would like to thank Kara Patterson, Julie Vaughan-Graham and Dina Brooks from the Department of Physical Therapy at the University of Toronto for their input in defining rehabilitation goals and outcomes.

REFERENCES

[1] B. S. Rupal, S. Rafique, A. Singla, E. Singla, M. Isaksson, and G. S. Virk, "Lower-limb exoskeletons: Research trends and regulatory guidelines in medical and non-medical applications," *Int. J. Adv. Robot. Syst.*, vol. 14, no. 6, pp. 1–27, Nov. 2017.

[2] B. Hobbs and P. Artemiadis, "A Review of Robot-Assisted Lower-Limb Stroke Therapy: Unexplored Paths and Future Directions in Gait Rehabilitation," *Front. Neurorobot.*, vol. 14, no. April, 2020.

[3] K. Lo, M. Stephenson, and C. Lockwood, "Effectiveness of robotic assisted rehabilitation for mobility and functional ability in adult stroke patients: a systematic review protocol," *JBI database of systematic reviews and implementation reports*, vol. 15, no. 12. pp. 3049–3091, Dec-2017.

[4] D. R. Louie and J. J. Eng, "Powered robotic exoskeletons in post-stroke rehabilitation of gait: A scoping review," *Journal of NeuroEngineering and Rehabilitation*, vol. 13, no. 1. BioMed Central, p. 53, 08-Dec-2016.

[5] S. Federici, F. Meloni, M. Bracalenti, and M. L. De Filippis, "The effectiveness of powered, active lower limb exoskeletons in neurorehabilitation: A systematic review," *NeuroRehabilitation*, vol. 37, no. 3, pp. 321–340, 2015.

[6] B. Chen *et al.*, "Recent developments and challenges of lower extremity exoskeletons," *Journal of Orthopaedic Translation*, vol. 5. 2016.

[7] R. Mendoza-Crespo, D. Torricelli, J. C. Huegel, J. L. Gordillo, J. L. Pons, and R. Soto, "An Adaptable Human-Like Gait Pattern Generator Derived From a Lower Limb Exoskeleton," *Front. Robot. AI*, vol. 6, p. 36, May 2019.

[8] J. Hong, C. Chun, S.-J. Kim, and F. C. Park, "Gaussian Process Trajectory Learning and Synthesis of Individualized Gait Motions," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 27, no. 6, pp. 1236–1245, Jun. 2019.

[9] G. Lv, H. Zhu, and R. D. Gregg, "On the design and control of highly backdrivable lower-limb exoskeletons: A discussion of past and ongoing work," *IEEE Control Syst.*, vol. 38, no. 6, pp. 88–113, Dec. 2018.

[10] G. Wu, C. Wang, X. Wu, Z. Wang, Y. Ma, and T. Zhang, "Gait Phase Prediction for Lower Limb Exoskeleton Robots," in *2016 IEEE International Conference on Information and Automation*, 2016, pp. 19–24.

[11] T. P. Luu, K. H. Low, X. Qu, H. B. Lim, and K. H. Hoon, "An individual-specific gait pattern prediction model based on generalized regression neural networks," *Gait Posture*, vol. 39, no. 1, pp. 443–448, Jan. 2014.

[12] F. Horst, S. Lapuschkin, W. Samek, K.-R. Müller, and W. I. Schöllhorn, "Explaining the unique nature of individual gait patterns with deep learning," *Sci. Rep.*, vol. 9, no. 1, p. 2391, Dec. 2019.

[13] H. B. Lim, Trieu Phat Luu, K. H. Hoon, and K. H. Low, "Natural gait parameters prediction for gait rehabilitation via artificial neural network," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 5398–5403.

[14] M. R. Tucker *et al.*, "Control strategies for active lower extremity prosthetics and orthotics: A review," *Journal of NeuroEngineering and Rehabilitation*, vol. 12, no. 1. 2015.

[15] T. Yan, M. Cempini, M. Oddo, and N. Vitiello, "Review of assistive strategies in powered lower-limb orthoses and exoskeletons," *Rob. Auton. Syst.*, vol. 64, pp. 120–136, 2015.

[16] B. Brahmi, M. Saad, C. O. Luna, P. S. Archambault, and M. H. Rahman, "Passive and active rehabilitation control of human upper-limb exoskeleton robot with dynamic uncertainties," *Robotica*, vol. 36, no. 11, pp. 1757–1779, Nov. 2018.

[17] Y. Long, Z. J. Du, W. D. Wang, and W. Dong, "Robust Sliding Mode Control Based on GA Optimization and CMAC Compensation for Lower Limb Exoskeleton," *Appl. Bionics Biomech.*, vol. 2016, 2016.

[18] F. Sado, H. J. Yap, R. Ariffin, R. A. R. Ghazilla, and N. Ahmad, "Exoskeleton robot control for synchronous walking assistance in repetitive manual handling works based on dual unscented Kalman filter," *PLoS One*, vol. 13, no. 7, 2018.

[19] D. Sanz-Merodio, M. Cestari, J. C. Arevalo, X. A. Carrillo, and E. Garcia, "Generation and control of adaptive gaits in lower-limb exoskeletons for motion assistance," *Adv. Robot.*, vol. 28, no. 5, pp. 329–338, Mar. 2014.

[20] S. K. Banala, S. K. Agrawal, S. H. Kim, and J. P. Scholz, "Novel gait adaptation and neuromotor training results using an active leg exoskeleton," *IEEE/ASME Trans. Mechatronics*, vol. 15, no. 2, pp. 216–225, 2010.

[21] X. Wang, X. Li, J. Wang, X. Fang, and X. Zhu, "Data-driven model-free adaptive sliding mode control for the multi degree-of-freedom robotic exoskeleton," *Inf. Sci. (Ny).*, vol. 327, pp. 246–257, Jan. 2016.

[22] J. Hwangbo *et al.*, "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, 2019.

[23] A. Rajeswaran *et al.*, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations," *ArXiv1709.10087*, 2018.

[24] Z. Qu *et al.*, "Research on Fuzzy Adaptive Impedance Control of Lower Extremity Exoskeleton," in *Proc. of 2019 IEEE International Conference*

on Mechatronics and Automation, 2019, pp. 939–944.

[25] Y. Yuan, Z. Li, T. Zhao, and D. Gan, "DMP-based Motion Generation for a Walking Exoskeleton Robot Using Reinforcement Learning," *IEEE Trans. Ind. Electron.*, vol. PP, no. c, 2019.

[26] R. Huang, H. Cheng, J. Qiu, and J. Zhang, "Learning Physical Human-Robot Interaction With Coupled Cooperative Primitives for a Lower Exoskeleton," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 4, pp. 1–9, 2019.

[27] V. Pong, S. Gu, M. Dalal, and S. Levine, "Temporal difference models: Model-free deep RL for model-based control," *6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc.*, pp. 1–14, 2018.

[28] G. Bingjing, H. Jianhai, L. Xiangpan, and Y. Lin, "Human–robot interactive control based on reinforcement learning for gait rehabilitation training robot," *Int. J. Adv. Robot. Syst.*, vol. 16, no. 2, Mar. 2019.

[29] Y. Zhang, S. Li, K. J. Nolan, and D. Zanotto, "Adaptive Assist-as-needed Control Based on Actor-Critic Reinforcement Learning," in *IEEE International Conference on Intelligent Robots and Systems*, 2019, pp. 4066–4071.

[30] M. Hamaya, T. Matsubara, T. Noda, T. Teramae, and J. Morimoto, "Learning assistive strategies from a few user-robot interactions: Model-based reinforcement learning approach," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 3346–3351, 2016.

[31] S. G. Khan, M. Tufail, S. H. Shah, and I. Ullah, "Reinforcement learning based compliance control of a robotic walk assist device," *Adv. Robot.*, pp. 1–12, Nov. 2019.

[32] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[33] S. Maggioni, N. Reinert, L. Lünenburger, and A. Melendez-Calderon, "An Adaptive and Hybrid End-Point/Joint Impedance Controller for Lower Limb Exoskeletons," *Front. Robot. AI*, vol. 5, p. 104, Oct. 2018.

[34] L. Wang, E. H. F. Van Asseldonk, and H. Van Der Kooij, "Model predictive control-based gait pattern generation for wearable exoskeletons," *IEEE Int. Conf. Rehabil. Robot.*, pp. 1–6, 2011.

[35] D. Lo Castro, C. H. Zhong, F. Braghin, and W. H. Liao, "Lower Limb Exoskeleton Control via Linear Quadratic Regulator and Disturbance Observer," in *2018 IEEE International Conference on Robotics and Biomimetics, ROBIO 2018*, 2018, pp. 1743–1748.

[36] O. Harib *et al.*, "Feedback Control of an Exoskeleton for Paraplegics: Toward Robustly Stable Hands-free Dynamic Walking," *ArXiv1802.08322*, Feb. 2018.

[37] S. Lee, M. Park, K. Lee, and J. Lee, "Scalable muscle-actuated human simulation and control," *ACM Trans. Graph.*, vol. 38, no. 4, 2019.

[38] X. Bin Peng, G. Berseth, K. Yin, and M. Van De Panne, "DeepLoco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Trans. Graph.*, vol. 36, no. 4, 2017.

[39] A. S. Anand, G. Zhao, H. Roth, and A. Seyfarth, "A deep reinforcement learning based approach towards generating human walking behavior with a neuromuscular model," *2019 IEEE-RAS 19th Int. Conf. Humanoid Robot.*, pp. 537–543, 2020.

[40] C. R. Gil, H. Calvo, and H. Sossa, "Learning an efficient gait cycle of a biped robot based on reinforcement learning and artificial neural networks," *Appl. Sci.*, vol. 9, no. 3, Feb. 2019.

[41] J. García and D. Shafie, "Teaching a humanoid robot to walk faster through Safe Reinforcement Learning," *Eng. Appl. Artif. Intell.*, vol. 88, Feb. 2020.

[42] C. Liu, A. Lonsberry, M. Nandor, M. Audu, A. Lonsberry, and R. Quinn, "Implementation of Deep Deterministic Policy Gradients for Controlling Dynamic Bipedal Walking," *Biomimetics*, vol. 4, no. 1, p. 28, 2019.

[43] V. C. V Kumar, S. Ha, G. Sawicki, and C. K. Liu, "Learning a Control Policy for Fall Prevention on an Assistive Walking Device," *ArXiv1909.10488*, 2019.

[44] D. Di Febbo *et al.*, "Reinforcement Learning Control of Functional Electrical Stimulation of the upper limb : a feasibility study," in *Annual Conference of the International Functional Electrical Stimulation Society*, 2018, pp. 111–114.

[45] M. Lyu, W. H. Chen, X. Ding, and J. Wang, "Knee exoskeleton enhanced with artificial intelligence to provide assistance-as-needed," *Rev. Sci. Instrum.*, vol. 90, no. 9, 2019.

[46] X. Zhang, H. Wang, Y. Tian, L. Peyrodie, and X. Wang, "Model-free based neural network control with time-delay estimation for lower extremity exoskeleton," *Neurocomputing*, vol. 272, pp. 178–188, Jan. 2018.

[47] P. Yang, J. Sun, J. Wang, G. Zhang, and Y. Zhang, "Model-free based back-stepping sliding mode control for wearable exoskeletons," in *25th IEEE International Conference on Automation and Computing*, 2019.

[48] Z. Li, J. Liu, Z. Huang, Y. Peng, H. Pu, and L. Ding, "Adaptive Impedance Control of Human-Robot Cooperation Using Reinforcement Learning," *IEEE Trans. Ind. Electron.*, vol. 64, no. 10, pp. 8013–8022, 2017.

[49] H. van Hasselt, "Reinforcement learning in continuous state and action spaces," in *Adaptation, Learning, and Optimization*, vol. 12, 2012, pp. 207–251.

[50] D. Di Febbo *et al.*, "Does Reinforcement Learning outperform PID in the control of FES-induced elbow flex-extension?," *2018 IEEE Int. Symp. Med. Meas. Appl. Proc.*, pp. 1–6, 2018.

[51] A. Singh, L. Yang, C. Finn, and S. Levine, "End-To-End Robotic Reinforcement Learning without Reward Engineering," *ArXiv1904.07854*, 2019.

[52] M. Plappert, "keras-rl," *GitHub*, 2016. [Online]. Available: https://github.com/keras-rl/keras-rl. [Accessed: 24-Apr-2020].

[53] Ł. Kidziński *et al.*, "Learning to Run Challenge: Synthesizing Physiologically Accurate Motion Using Deep Reinforcement Learning," in *Escalera S., Weimer M. (eds) The NIPS '17 Competition: Building Intelligent Systems. The Springer Series on Challenges in Machine Learning*, Springer, Cham, 2018, pp. 101–120.

[54] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the Brownian motion," *Phys. Rev.*, vol. 36, no. 5, pp. 823–841, Sep. 1930.

[55] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[56] G. Brockman *et al.*, "OpenAI Gym," *ArXiv1606.01540*, 2016.

[57] D. Thelen, A. Seth, F. C. Anderson, and S. L. Delp, "OpenSim Models Gait 2392 and 2354 Documentation," *SimTK*. [Online]. Available: https://simtk-confluence.stanford.edu:8443/display/OpenSim/Gait+2392+and+2354+Models. [Accessed: 09-May-2020].

[58] E. P. Grabke, K. Masani, and J. Andrysek, "Lower Limb Assistive Device Design Optimization Using Musculoskeletal Modeling: A Review," *J. Med. Device.*, vol. 13, no. 4, 2019.

[59] M. Khamar, M. Edrisi, and M. Zahiri, "Human-exoskeleton control simulation, kinetic and kinematic modeling and parameters extraction," *MethodsX*, vol. 6, pp. 1838–1846, 2019.

[60] D. Coll Pujals, "Simulation of the assistance of an exoskeleton on lower limbs joints using Opensim," Polytechnic University of Catalonia, 2017.

[61] M. A. Sherman, A. Seth, and S. L. Delp, "Simbody: Multibody dynamics for biomedical research," *Procedia IUTAM*, vol. 2, pp. 241–261, 2011.

[62] L. Rose, M. C. F. Bazzocchi, C. de Souza, J. Vaughan-Graham, K. Patterson, and G. Nejat, "A Framework for Mapping and Controlling Exoskeleton Gait Patterns in both Simulation and Real -World," in *Proc. of the 2020 Design of Medical Devices Conf.*, 2020.