ExoSwift: GPU-Parallel Exoskeleton Policy Learning and Distillation for Real-Time Embedded Deployment

***Abstract*— Musculoskeletal simulation can reduce physical experimentation in exoskeleton controller development, but the computational costs of human-exoskeleton physical simulation and real-time embedded inference remain important constraints. We present ExoSwift, a staged method that separates upstream musculoskeletal human-controller preparation from downstream exoskeleton policy learning and embedded execution. ExoSwift uses a frozen human controller for GPU-parallel downstream policy optimization and then distills the learned assistance policy for real-time microcontroller deployment. On a single RTX 5080, the GPU-resident trainer sustained 10,961 environment steps/s with 2,048 environments, corresponding to $13\times$ the peak measured throughput of the evaluated CPU implementations on the same machine. Across three random seeds, the mean convergence time ranged from 6.0 to 13.5 min across the four parallelism settings. Policy distillation reduced the parameter count by $15.2\times$; the student policy required 2.03 ms for bilateral inference on a Teensy 4.1. In treadmill experiments with six participants, the mean participant-level concordance correlation coefficient (CCC) for torque commands ranged from 0.91 to 0.94 across four walking speeds, with SDs of 0.03–0.04.**

I. Introduction

Powered exoskeletons have shown promise for assisting mobility and reducing effort [1]–[3], but determining *how* a device should assist remains expensive. Human-in-the-loop optimization tunes assistance parameters on the physical device [4]–[6], yet each evaluation requires substantial participant walking time and the search is limited to low-dimensional profile families. Data-driven approaches offer an alternative [7], [8], but supervised formulations still require large labeled datasets spanning users and locomotion conditions.

Simulation-based controller design shifts exoskeleton policy optimization from repeated human experiments to virtual interactions, where assistance can be evaluated using biomechanical quantities directly accessible in simulation. Kim et al. [9] use rigid-body simulation and DDPG to generate smooth locomotion transitions, but their model does not represent muscle activation. Yuan et al. [10] introduce a four-stage curriculum that progressively trains a musculoskeletal human actor and a bilateral exoskeleton actor, followed by deployment of the resulting controller on a hip exoskeleton for biomechanical and metabolic validation. ExoPlore [11] develops a human-aligned musculoskeletal simulation method, but rather than learning an exoskeleton policy through reinforcement learning, it optimizes the gain and delay parameters of a predefined delayed output-feedback controller [12], restricting assistance to a fixed torque-generation structure. Luo et al. [13] learn a hip exoskeleton controller in simulation and deploy it on hardware, although their study reports limited quantitative analysis of simulation-to-hardware differences. Park et al. [14] develop a policy-distillation framework that transfers a teacher with simulator-only privileged observations to a policy with deployable IMU inputs. Robbins et al. [15] introduce MyoAssist 1.0, a MuJoCo-based framework that composes musculoskeletal human models, assistive devices, and locomotion tasks and supports both reinforcement learning and structured controller optimization workflows. More recently, Yuan et al. [16] propose UniExo, which co-adapts a unified musculoskeletal human controller with an exoskeleton policy across walking, turning, running, backward walking, and transitions using GPU-based MuJoCo Warp simulation.

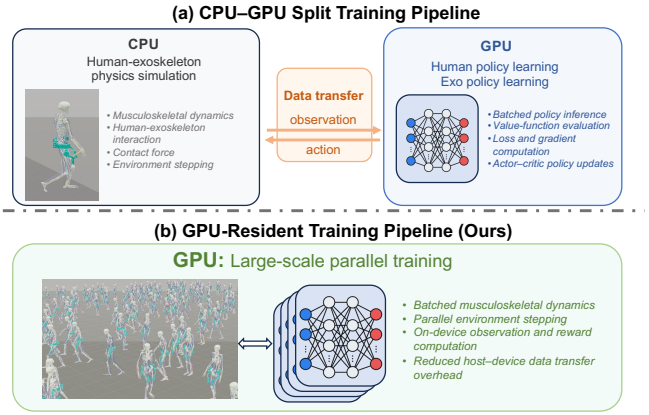

Fig. 1. ExoSwift keeps human–exoskeleton simulation and policy learning on the GPU, supporting large-scale parallel training. **(a)** In a CPU–GPU split pipeline, CPU-based physics simulation limits rollout parallelism across large batches of musculoskeletal environments; policy learning on the GPU additionally requires observations and actions to be transferred at each simulation step. **(b)** In the ExoSwift training pipeline, batched simulation and policy learning are both GPU-resident.

Simulation-based exoskeleton studies have learned assistance policies and deployed them on hardware, but two efficiency issues remain unresolved. First, it remains unclear how large-scale parallel training affects exoskeleton assistance policy optimization time. Many existing implementations use CPU-based simulation for human–exoskeleton interaction and rollout collection, limiting the available rollout parallelism, as shown in Fig. 1. Second, RL policies often contain hundreds of thousands of parameters and therefore rely on NVIDIA Jetson or Raspberry Pi devices for real-time deployment. Their hardware size and cost create a need for compact policies that can run directly on microcontrollers

All experimental procedures were approved by the Institutional Review Board (IRB) of the authors' institution.

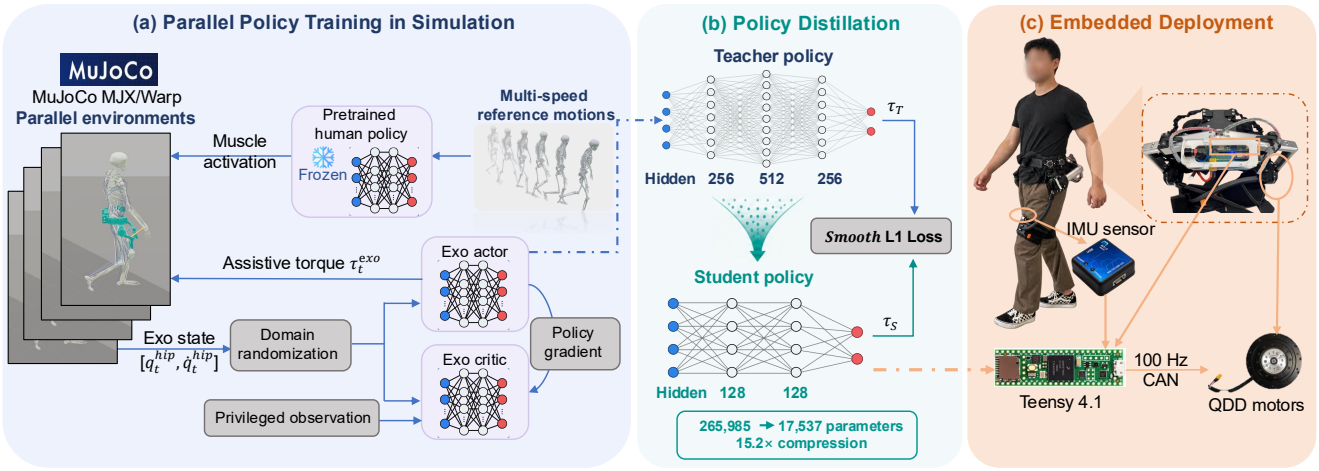

Fig. 2. ExoSwift trains one assistance policy across four walking speeds using GPU-resident parallel simulation and deploys the distilled policy on a microcontroller. **(a)** A frozen, perturbation-adapted musculoskeletal human controller tracks four walking speeds while the exoskeleton actor–critic learns assistive hip torques. **(b)** Policy distillation transfers the teacher policy's torque mapping to a compact student policy, reducing the parameter count from 265,985 to 17,537 (15.2×). **(c)** The student policy performs bilateral inference in 2.03 ms on a Teensy 4.1, occupying approximately 20 % of the 100 Hz control period and enabling IMU sensing, neural-network inference, and QDD motor control on the same microcontroller.

of portable exoskeletons. Table I summarizes the simulation and deployment platforms used in related studies.

These issues in downstream exoskeleton policy training efficiency and real-time microcontroller deployment motivate two research questions: (1) With a pretrained musculoskeletal human controller frozen, how quickly can GPU-parallel training optimize a downstream exoskeleton policy? (2) Can policy distillation preserve the learned torque mapping while meeting real-time microcontroller constraints? These questions differ from the focus of UniExo [16]. UniExo develops a unified musculoskeletal human controller and studies human–exoskeleton co-adaptation across multiple locomotion skills. ExoSwift instead holds a pretrained human controller fixed to isolate how quickly a downstream exoskeleton policy can be trained, and then evaluates whether that policy can be compressed for microcontroller execution.

To address these questions, ExoSwift uses the GPU-resident MuscleMimic framework [17] as the foundation for musculoskeletal human simulation and extends it to downstream exoskeleton policy learning. Starting from a pretrained human controller, we apply a hip-torque perturbation curriculum; the resulting human controller is then frozen and reused for downstream exoskeleton policy learning. The reported downstream training times exclude the upstream cost of preparing the human controller. ExoSwift then distills the learned teacher exoskeleton policy into a compact student policy for execution on a Teensy 4.1 microcontroller.

The contributions of ExoSwift are twofold. First, ExoSwift isolates downstream exoskeleton policy learning around a perturbation-adapted, frozen MuscleMimic controller and quantifies how GPU parallelism affects exoskeleton policy training speed and time. Second, we evaluate whether policy distillation can reduce model size and inference latency to meet real-time microcontroller constraints while preserving the learned torque mapping of the teacher policy.

TABLE I

COMPARISON OF SIMULATION-BASED EXOSKELETON CONTROL STUDIES.

Study	Human model	Platform	Deployment device
Luo et al. [13]	Muscle	DART (CPU)	Raspberry Pi 4
Kim et al. [9]	Rigid	Isaac Gym (GPU)	Jetson Orin Nano
Zhou et al. [18]	Rigid	Isaac Lab (GPU)	Host PC
Barati et al. [19]	Muscle	Hyfydy (CPU)	Portenta X8
Yuan et al. [10]	Muscle	MuJoCo (CPU)	Raspberry Pi 5
Park et al. [14]	Muscle	Hyfydy (CPU)	Jetson Orin Nano
Leem et al. [11]	Muscle	DART (CPU)	–
Robbins et al. [15]	Muscle	MuJoCo (CPU)	–
Yuan et al. [16]	Muscle	MuJoCo Warp (GPU)	Raspberry Pi 5
Ours	**Muscle**	**MuJoCo MJX/Warp (GPU)**	**Teensy 4.1**

II. Simulation Platform and Human-Controller Adaptation

ExoSwift uses the GPU-resident MuscleMimic framework [17] as the foundation for musculoskeletal human simulation. MuscleMimic provides the musculoskeletal model, batched simulator, imitation objective, and pretrained generalist human controller. ExoSwift extends this foundation to human–exoskeleton interaction and downstream exoskeleton assistance policy learning with a perturbation-adapted, frozen human controller.

A. Musculoskeletal model and exoskeleton model

The musculoskeletal environments are implemented in JAX using MuJoCo MJX and the MuJoCo Warp backend. MJX provides the JAX-compatible batched simulation interface, while the MuJoCo Warp backend evaluates the contact-rich musculoskeletal dynamics in parallel on the NVIDIA GPU. The musculoskeletal model is a free-root, full-body model actuated by Hill-type muscle-tendon units [20], with finger muscles disabled, resulting in 83 joints and 354 active muscle actuators.

A hip exoskeleton is attached to the pelvis and thighs. The device mass and inertia are incorporated into the composite inertial properties of the corresponding human segments. The modeled device has a total mass of 2.79 kg without the wearable and battery, comprising a 1.99 kg pelvis-mounted base, two 0.095 kg proximal links, and two 0.306 kg thigh-mounted components. The device contributes one sagittal-plane actuation degree of freedom at each hip.

B. Perturbation Adaptation of the Human Controller

Although the pretrained human controller captures diverse walking behaviors, its original training does not explicitly include external torques applied directly at the hip joints. This omission matters during early exoskeleton-policy training, when the randomly initialized device policy can produce torques that disrupt the human gait and hinder the learning of useful assistance. We therefore initialize the human controller from the pretrained checkpoint and adapt it for 3×10^8 simulation steps using PPO [21], a DeepMimic-style imitation reward [22], and a torque-perturbation curriculum. This stage required approximately 14 hours on our training system. Specifically, temporally correlated bilateral hip torques are generated, with their maximum amplitude linearly increased from 0 to $10\,\mathrm{N\,m}$ over this additional training stage. The resulting human controller is subsequently frozen for exoskeleton policy learning. This curriculum is intended to keep early exoskeleton exploration closer to the nominal gait and provide a more informative learning signal for assistance.

III. Exoskeleton Policy Learning

Once pretrained and frozen, the same human controller can be reused across exoskeleton designs and reference motions within its supported motion distribution. In this work, we train a single hip exoskeleton policy using PPO across four walking speeds (0.6, 0.8, 1.0, and 1.2 m/s), as shown in Fig. 2.

A. Exoskeleton Observation and Action

The exoskeleton actor uses only the corresponding leg's hip flexion angle and angular velocity as observations. For each leg $i \in \{r, l\}$,

$$\mathbf{o}_t^{\mathrm{exo},i} = \left[q_{h,t}^i, \dot{q}_{h,t}^i\right]^\mathsf{T}, \tag{1}$$

where $q_{h,t}^i$ and $\dot{q}_{h,t}^i$ denote the hip flexion angle and angular velocity of leg i at time step t, respectively. During training, the critic instead receives the normalized 2,418-dimensional privileged environment observation, comprising full-body joint kinematics, muscle states, foot-contact signals, and reference-motion features. This asymmetric actor–critic formulation uses privileged simulation information for value estimation without introducing simulator-only inputs into the deployed actor.

The same actor is evaluated independently for each leg:

$$a_t^i = \pi_\theta\left(q_{h,t}^i, \dot{q}_{h,t}^i\right), \qquad i \in \{r, l\}. \tag{2}$$

The network parameters are shared across legs, while each torque output depends on the corresponding leg's local kinematic state. The actor network is a multilayer perceptron (MLP) with three hidden layers of 256, 512, and 256 units, using SiLU activations and LayerNorm. The normalized policy outputs are converted into assistive hip torques as

$$\tau_t^i = \tau_{\max}\,\mathrm{clip}(a_t^i, -1, 1), \qquad \tau_{\max} = 15\,\mathrm{N\,m}. \tag{3}$$

Before being applied to the simulated hip joints, the commanded torques pass through a second-order Butterworth low-pass filter with a cutoff frequency of 3 Hz.

B. Domain Randomization of IMU-Based Policy Inputs

To expose the exoskeleton policy to IMU discrepancies and drift during deployment [23], domain randomization is applied to the hip-angle and angular-velocity observations. The frozen human controller receives the unperturbed simulation observation, whereas the randomized observation is passed to the exoskeleton actor and critic.

For each episode and each leg, the measured hip angle is modeled as

$$\tilde{q}_{h,t}^i = g^i q_{h,t}^i + b^i, \tag{4}$$

where

$$b^i \sim \mathcal{U}(-10^\circ, 10^\circ), \qquad g^i \sim \mathcal{U}(0.85, 1.15). \tag{5}$$

The offset b^i represents calibration error, while the gain g^i accounts for differences in sensor mounting and the resulting angle scale. Both quantities remain constant within an episode.

To approximate the short-term temporal correlation measured in the IMU signals, we add a length-K moving average of independent Gaussian samples to the hip angular velocity:

$$\tilde{\dot{q}}_{h,t}^i = \dot{q}_{h,t}^i + \frac{\sigma_{\dot{q}}}{\sqrt{K}} \sum_{k=0}^{K-1} \epsilon_{t-k}^i, \qquad \epsilon_{t-k}^i \sim \mathcal{N}(0,1), \tag{6}$$

where $\sigma_{\dot{q}} = 43^\circ/\mathrm{s}$ and $K = 4$.

C. Exoskeleton Reward Function

We retain the human motion-tracking reward r_t^{track} from MuscleMimic [17] and introduce an exoskeleton-specific assistance reward r_t^{exo} for downstream policy learning. The exoskeleton reward comprises three terms:

$$\begin{aligned} r_t &= r_t^{\mathrm{track}} + r_t^{\mathrm{exo}}, \\ r_t^{\mathrm{exo}} &= r_t^{\mathrm{bio}} + r_t^{\mathrm{power}} + r_t^{\mathrm{reg}}, \end{aligned} \tag{7}$$

where r_t^{bio} is the biological work proxy reward, r_t^{power} is the assistive mechanical power reward, and r_t^{reg} is the assistive torque regularization reward.

a) Biological work proxy reward: For each leg $i \in \{r, l\}$, the biological joint power is

$$P_{\mathrm{bio},t}^i = M_{\mathrm{bio},t}^i \dot{q}_{h,t}^i, \tag{8}$$

where $M_{\mathrm{bio},t}^i$ is the biological hip moment of leg i. The biological effort term uses a biological work proxy in which

negative biological work is assigned a smaller weight than positive biological work, motivated by [24], [25].

$$r_t^{\text{bio}} = -\frac{\lambda_{\text{bio}}}{2P_{\text{bio}}} \sum_{i\in\{r,l\}} \left[\max\left(P^i_{\text{bio},t}, 0\right) + \alpha_{\text{neg}} \max\left(-P^i_{\text{bio},t}, 0\right)\right] \tag{9}$$

where $\lambda_{\text{bio}} = 0.15$, $\alpha_{\text{neg}} = 0.21$, and $P_{\text{bio}} = 60\,\text{W}$.

b) Assistive mechanical power reward: The mechanical power delivered by the exoskeleton to each leg is

$$P^i_{\text{exo},t} = \tau^i_t \dot{q}^i_{h,t}. \tag{10}$$

Positive mechanical power is rewarded, whereas negative work is penalized as follows:

$$\phi^i_t = \max\left(P^i_{\text{exo},t}, 0\right) - \beta \max\left(-P^i_{\text{exo},t}, 0\right), \tag{11}$$

$$r_t^{\text{power}} = \lambda_{\text{power}} g_t \tanh\left(\frac{1}{P_{\text{exo}}} \sum_{i\in\{r,l\}} \phi^i_t\right), \tag{12}$$

where $\lambda_{\text{power}} = 0.15$, $\beta = 1.5$, and $P_{\text{exo}} = 45\,\text{W}$. The tracking gate is defined as $g_t = (r^q_t r^{\text{root}}_t)^\kappa$, where r^q_t and r^{root}_t are the joint angle and root position tracking rewards in MuscleMimic [17]. We set $\kappa = 1$. Thus, the mechanical-power term remains large when the assisted motion closely tracks the reference and decreases as the tracking error increases.

c) Assistive torque regularization reward: The regularization reward discourages actuator saturation and large torques at low joint velocities:

$$r_t^{\text{reg}} = -\frac{1}{2} \sum_{i\in\{r,l\}} \left[\lambda_{\text{reg}}\, w^i_t \left(\frac{\tau^i_t}{\tau_{\max}}\right)^2 + \lambda_{\text{sat}} \left(\frac{\tau^i_t}{\tau_{\max}}\right)^6\right], \tag{13}$$

$$w^i_t = \max\left(1 - \frac{|\dot{q}^i_{h,t}|}{\omega_{\max}}, w_{\min}\right), \tag{14}$$

where $\lambda_{\text{reg}} = 0.10$, $\lambda_{\text{sat}} = 0.20$, $\omega_{\max} = 4\,\text{rad/s}$, and $w_{\min} = 0.05$. The two terms act on different parts of the torque profile. The velocity weight w^i_t penalizes torque most strongly near velocity reversals and gradually reduces the penalty as the joint speed increases. The lower bound $w_{\min}$ retains a nonzero torque penalty at high velocities. This term therefore shapes the entire torque trajectory, and the policy is encouraged to reduce low-power torque and concentrate assistance during phases in which joint motion can convert the applied torque into mechanical work. The second term is negligible below roughly half of $\tau_{\max}$ and rises steeply as the command approaches the limit, acting only as a barrier against actuator saturation.

IV. Policy Distillation for Embedded Deployment

For portable exoskeletons, neural-network policies should execute on compact embedded processors while sharing each real-time control period with sensing, motor control, and communication. The teacher policy produced by the ExoSwift training pipeline is sized for RL optimization rather than embedded inference efficiency. We therefore use offline policy distillation to train a lightweight student policy to reproduce the teacher's torque outputs for real-time execution on a Teensy 4.1 microcontroller.

TABLE II

Distillation brings bilateral policy inference within the 10-ms control period on a Teensy 4.1.

Metric	Teacher policy	Student policy
Parameters	265,985	17,537
Weight storage	1,039.0 KiB	68.5 KiB
Bilateral inference time	31.44 ms	2.03 ms
Fraction of 10 ms period	314%	20%

A. Teacher and Student Policies

The trained exoskeleton policy serves as the teacher policy. Because the policy is applied independently to each leg with shared weights, each network maps a two-dimensional input, comprising the hip angle and angular velocity, to one assistive torque output. The teacher policy uses three hidden layers with 256, 512, and 256 units, whereas the student policy π_{S} uses two hidden layers of 128 units. Each hidden layer contains a fully connected layer, LayerNorm, and a SiLU activation. The student policy retains the teacher policy's input normalization, torque scaling, and output limits. The teacher and student policies contain 265,985 and 17,537 parameters, respectively, corresponding to a $15.2\times$ compression and a 93.4% reduction.

B. Offline Policy Distillation

The trained exoskeleton policy is distilled through offline regression from the teacher policy to the student policy. No additional simulation rollouts or experimental data are used. We uniformly sample 400,000 inputs over the normalized domain corresponding to hip angles of $\pm 60^\circ$ and angular velocities of $\pm 500^\circ/\text{s}$, and add 200,000 Gaussian samples concentrated around the walking operating region. The teacher policy evaluates all 600,000 inputs to generate torque labels clipped to $\pm 15\,\text{N m}$. The parameters of the student policy are optimized using

$$\theta^*_{\text{S}} = \arg\min_{\theta_{\text{S}}} \frac{1}{N} \sum_{n=1}^{N} \text{SmoothL1}_\beta\left(\tau_{\text{S},n} - \tau_{\text{T},n}\right), \tag{15}$$

where $\tau_{\text{S},n}$ and $\tau_{\text{T},n}$ denote the clipped outputs of the student and teacher policies, respectively, and $\beta = 0.5$.

C. Embedded Implementation

The distilled student policy is exported as 32-bit floating-point C arrays and stored in the program flash of a Teensy 4.1 microcontroller. The policy weights occupy approximately 68.5 KiB and are accessed directly from flash, while only two 128-element intermediate buffers are required during inference. The Teensy serves as the real-time control processor and executes the complete real-time control loop at 100 Hz. At each control cycle, the bilateral hip angles and angular velocities are obtained from the hip IMUs. The

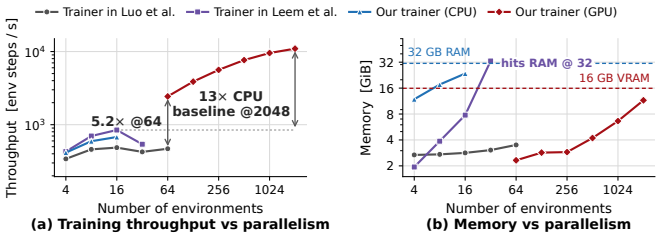

Fig. 3. The GPU-resident ExoSwift trainer reaches 10,961 environment steps/s with 2,048 parallel environments, corresponding to 13× the peak measured throughput of the three evaluated CPU implementations on the same machine. **(a)** End-to-end training throughput, including rollout collection and policy updates. **(b)** CPU RAM or GPU VRAM usage.

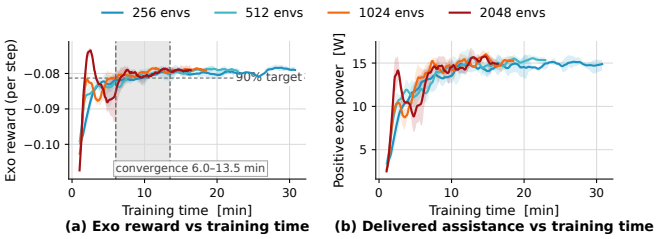

Fig. 4. All tested environment counts reached similar final reward and positive exoskeleton power across three random seeds. **(a)** Mean exoskeleton reward; the dashed line marks the 90% target. **(b)** Mean positive exoskeleton power. Curves and shaded bands show the across-seed mean and range, respectively.

TABLE III

TRAINING HYPERPARAMETERS ACROSS PARALLEL ENVIRONMENT NUMBERS.

Hyperparameter	**Parallel environments N**			
	256	**512**	**1024**	**2048**
Rollout horizon T	160	80	40	20
Update batch $N \times T$	40,960	40,960	40,960	40,960
Minibatches per update	256	256	256	256
Total environment steps	10^7			
Minibatch size	160			
Learning rate	$2 \times 10^{-4} \rightarrow 2 \times 10^{-5}$ (linear)			
Discount factor γ	0.99			
GAE parameter λ	0.95			
PPO clipping coefficient	0.2			

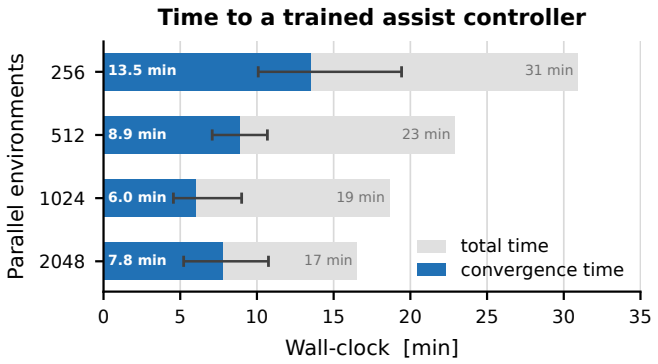

Fig. 5. Increasing parallelism from 256 to 2,048 environments reduced the wall-clock time for the full 10^7-step training budget from 31 to 17 min. Across three random seeds, mean convergence time ranged from 6.0 to 13.5 min. Blue bars and error bars show the mean and range, respectively; gray bars show the wall-clock time for the full 10^7 step budget.

output assistive torques pass through a second-order 3 Hz Butterworth filter and are then limited by the prescribed torque bound before being transmitted to the QDD motors via CAN.

As summarized in TABLE II, the teacher policy requires 31.44 ms for one bilateral inference, corresponding to 314% of the 10-ms control period. In comparison, the distilled student policy requires 2.03 ms, or approximately 20% of the period. Distillation therefore reduces the parameter count by a factor of 15.2 and the measured inference time by a factor of 15.5. Inference with the student policy leaves approximately 7.97 ms of each period for sensor processing, motor communication, and other control tasks on the same microcontroller.

V. EXPERIMENTS AND RESULTS

We evaluate ExoSwift against the two research questions defined in the Introduction. Section V-A quantifies how environment parallelism affects downstream training throughput and optimization time. Section V-B evaluates whether the learned policy achieves its intended assistance objective in simulation. Section V-C evaluates output fidelity between the teacher and student policies, microcontroller inference time, and participant-level simulation-to-deployment agreement in treadmill experiments with six participants.

A. Training Efficiency

We compare the GPU-resident trainer with three CPU-based exoskeleton policy-learning implementations: a matched CPU version of our trainer, the implementation used in Luo et al. [13], and the implementation used in Leem et al. [11]. Our trainer (CPU) uses the same musculoskeletal environment, policy architecture, PPO configuration, and training implementation as the GPU-resident trainer; only the computational backend differs. The other two CPU implementations were built from source. All four implementations were evaluated on the same machine, equipped with a 20-core Intel Core Ultra 7 CPU, 32 GB of RAM, and an RTX 5080 GPU. Throughput measurements cover the complete training loop, including rollout collection and policy updates, rather than physics stepping alone. Fig. 3 shows that the evaluated CPU implementations peak below 850 steps/s, whereas the GPU-resident trainer scales from 2,435 steps/s with 64 environments to 10,961 steps/s with 2,048 environments. This corresponds to 13× the peak measured throughput of the evaluated CPU implementations. At 2,048 environments, the GPU-resident trainer uses 72% of the 16-GB GPU memory.

To compare different levels of environment parallelism under the same optimization budget, we jointly scale the number of environments N and rollout horizon T such that $NT = 40{,}960$ transitions per PPO update. The minibatch size, number of minibatches, and total environment steps budget are also fixed. Consequently, all settings use the same number of PPO iterations and parameter updates. The training hyperparameters are listed in TABLE III. For each

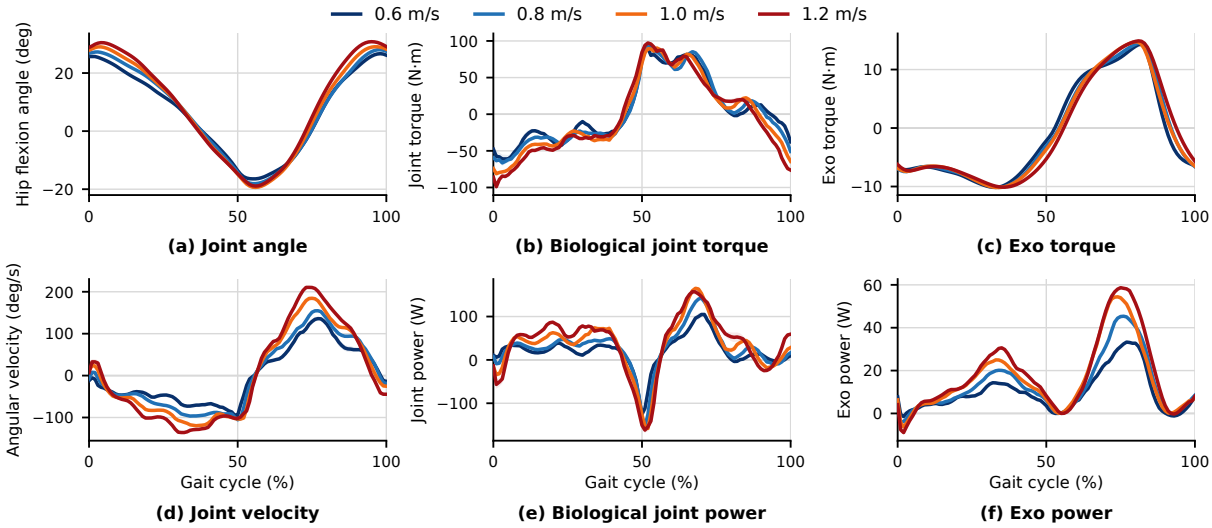

Fig. 6. In simulation, the same policy provides extension assistance during stance and flexion assistance during swing at all four trained walking speeds, while positive exoskeleton power increases with speed. **(a, d)** Hip angle and angular velocity in the No Exo condition; **(b, e)** simulated biological hip torque and power in the No Exo condition; and **(c, f)** learned assistive torque and mechanical power in the Assist On condition.

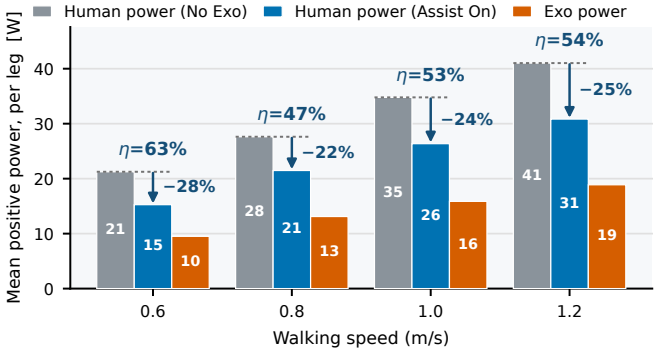

Fig. 7. Assistance reduces simulated positive biological hip power by 22–28% across the four walking speeds while the device delivers 9.5–18.9 W of positive mechanical power per leg. Bars show biological hip power in the No Exo and Assist On conditions. Positive power is averaged over the total rollout time. Labels report the mechanical conversion ratio $\eta = (P^{+}_{\mathrm{bio,NoExo}} - P^{+}_{\mathrm{bio,AssistOn}})/P^{+}_{\mathrm{device}}$.

seed, we define convergence time as the earliest wall-clock time after which the exoskeleton reward remains above a common threshold corresponding to 90% of the reward improvement from the initial value to the final reference value. Figs. 4 and 5 summarize three random seeds for 256, 512, 1,024, and 2,048 parallel environments under the same budget of 10^7 environment steps. The mean convergence times were 13.5, 8.9, 6.0, and 7.8 min for 256, 512, 1,024, and 2,048 parallel environments. The shorter rollout at 2,048 environments ($T = 20$) may reduce the quality of the GAE estimates [27], partly offsetting the increased sampling throughput.

B. Assistive Policy Evaluation in Simulation

We next evaluate the exoskeleton controller trained jointly at four walking speeds (0.6, 0.8, 1.0, and 1.2 m/s). Fig. 6 summarizes the gait-cycle-averaged hip kinematics and kinetics. Hip kinematics followed similar gait-cycle patterns across speeds, while simulated biological torque and power increased with walking speed. The learned controller produced assistive torque profiles with similar stance-extension and swing-flexion structure across the four speeds, with torque crossing zero near the reversal of hip motion. Fig. 7 shows that the policy reduced simulated positive biological hip power by 22–28% while delivering 9.5–18.9 W of positive mechanical power per leg, corresponding to a mechanical conversion ratio of 47–63%.

C. Deployment Protocol and Results

The experimental protocol comprised four treadmill walking speeds (0.6, 0.8, 1.0, and 1.2 m/s), with each speed held constant for at least 60 s. The four steady-speed segments were presented consecutively without rest, and treadmill speed was varied between adjacent segments. This protocol evaluated the controller at four consecutive steady walking speeds. Six healthy adults participated ($n = 6$). The participants' demographics were as follows: age, 22.83 ± 2.14 years; weight, 78.87 ± 15.87 kg; height, 177.17 ± 5.85 cm (mean $\pm$ standard deviation).

Fig. 8 compares simulated gait-cycle profiles with those recorded during deployment. For each participant and walking speed, gait cycles from both legs were pooled to obtain a participant-specific mean curve. CCC relative to simulation was calculated separately for hip angle, angular velocity, and torque command before being summarized as the mean and SD across six participants. Across the four speeds, the mean participant-level CCC ranged from 0.82 to 0.85 for hip angle, from 0.75 to 0.86 for angular velocity, and from 0.91 to 0.94 for torque command. The higher mean torque-command CCC at each speed shows that the learned policy produced more simulation-consistent torque profiles than the measured hip kinematics across the tested conditions.

To quantify output fidelity after distillation, we compared the replayed output of the teacher policy with the output of the student policy measured on the Teensy 4.1 over the

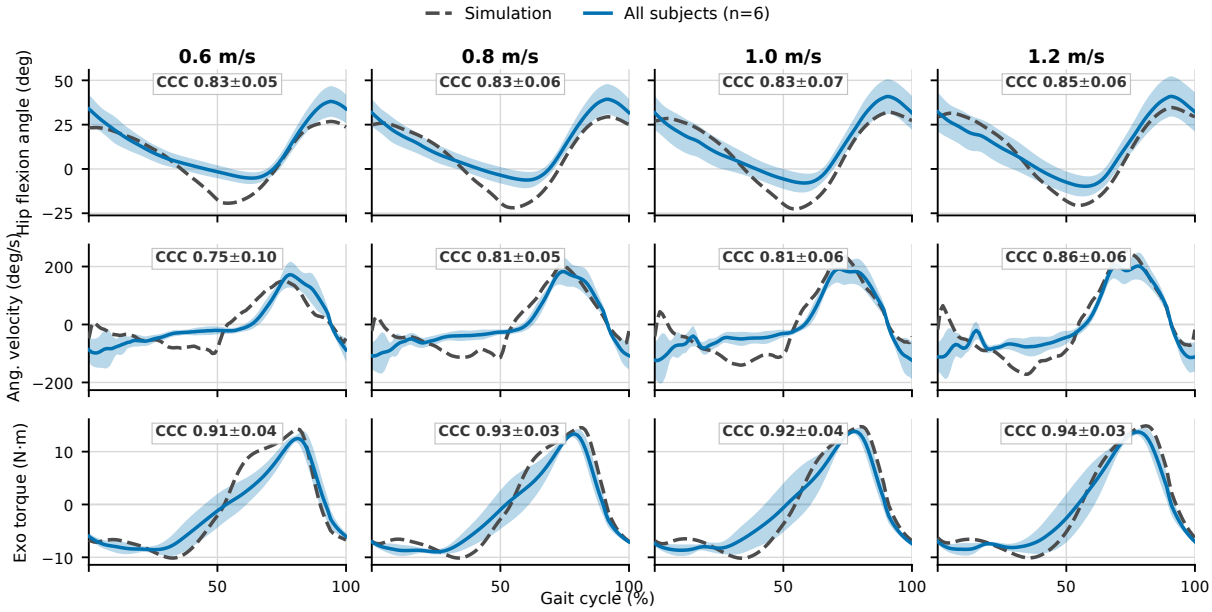

Fig. 8. Torque commands showed higher participant-level simulation-to-deployment agreement than hip angle and angular velocity at all four walking speeds. Lin's CCC [26] was computed per participant and summarized as mean ± SD ($n = 6$). Dashed gray curves show simulation; blue curves and shading show the participant mean ± SD, aligned to the simulation-derived heel strike (0%).

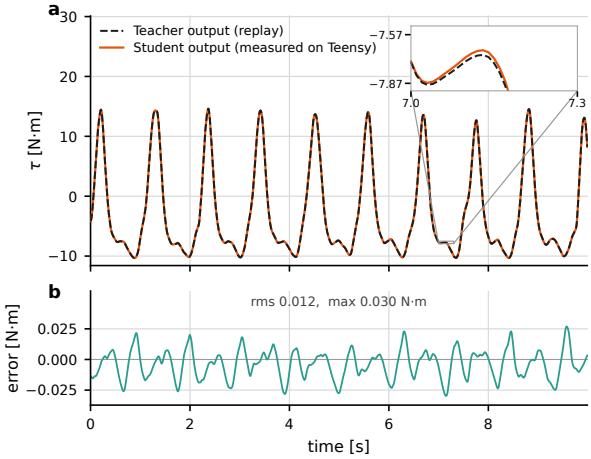

Fig. 9. Policy distillation preserved the teacher policy's torque output during embedded inference. The curve is from 10-s walking segment from Participant 1 at 1.2 m/s. (a) Teacher output obtained from replay and student output measured on the Teensy 4.1; (b) student–teacher output error. The RMS error was 0.012 N·m, and the maximum absolute error was 0.030 N·m.

same 10-s sequence. As shown in Fig. 9, the two torque profiles closely overlap throughout the evaluated sequence. The student–teacher error had an RMS value of 0.012 N·m and a maximum absolute value of 0.030 N·m. These results show that the 15.2× parameter reduction retained the teacher policy's torque output while satisfying the embedded inference constraint.

TABLE IV summarizes deployment metrics for torque command and command-derived mechanical power. Command-derived instantaneous mechanical power was calculated as the torque command multiplied by the IMU-derived hip angular velocity. For each participant and walking speed, gait cycles from both legs were pooled. Mean positive and negative command-derived power were obtained by separately averaging the positive and negative portions over the complete gait cycle. The positive-power ratio was defined as mean positive command-derived power divided by the sum of mean positive command-derived power and the magnitude of mean negative command-derived power. Each metric was first calculated for each participant and then summarized as the mean and standard deviation across the six participants. As walking speed increased from 0.6 to 1.2 m/s, RMS torque command increased from 7.54 to 8.29 N·m, while mean positive command-derived power increased from 9.32 to 14.53 W. Across the four speeds, 98.18–98.43% of the command-derived mechanical power was positive, indicating that the deployed policy preserved the predominantly positive-power behavior encouraged during simulation training.

VI. Limitations and Conclusion

A. Limitations

This study has three main limitations. First, ExoSwift relies on a pretrained, musculoskeletal human controller, which remains computationally expensive to obtain. The reported training efficiency therefore applies only to downstream exoskeleton policy optimization once a suitable human controller is available. Second, the human experiments evaluated real-time deployment and torque-command consistency but did not include EMG, metabolic measurements, or other direct physiological measures of assistance efficacy. The reductions in biological hip power reported here are simulation predictions and should not be interpreted as experimentally measured reductions in muscle effort. Third, deployment mechanical power was command-derived from torque commands and IMU-derived hip angular velocity rather than measured interaction torque. It was lower than simulated

TABLE IV

PARTICIPANT-LEVEL TORQUE COMMAND AND COMMAND-DERIVED MECHANICAL POWER AT FOUR WALKING SPEEDS(N = 6).

	Walking speed (m/s)			
Metric	0.6	0.8	1.0	1.2
τ_{RMS} (N·m)	7.54 (0.21)	7.83 (0.28)	8.02 (0.16)	8.29 (0.16)
$\overline{P}_{+}$ (W)	9.32 (2.02)	10.74 (2.22)	12.48 (2.20)	14.53 (2.44)
$\overline{P}_{-}$ (W)	−0.15 (0.08)	−0.16 (0.10)	−0.21 (0.10)	−0.22 (0.10)
ρ_{+} (%)	98.18 (1.59)	98.36 (1.52)	98.28 (0.93)	98.43 (1.05)

Note: τ_{RMS} denotes the RMS torque command; $\overline{P}_{+}$ and $\overline{P}_{-}$ denote mean positive and negative command-derived mechanical power for one leg, respectively. ρ_{+} was first calculated for each participant as the ratio of mean positive command-derived power to total absolute command-derived power. The table reports the mean (SD) of the participant-specific ratios.

device power, a mismatch that may reflect differences in hip angular velocity between simulated and measured walking.

B. Conclusion

ExoSwift establishes a staged formulation in which a perturbation-adapted musculoskeletal human controller is frozen and reused, isolating downstream exoskeleton policy learning from upstream controller preparation. GPU-resident scaling increases rollout throughput and shortens the fixed-budget training time, while non-monotonic convergence times show that throughput alone does not determine optimization speed. Policy distillation separates the network used for RL optimization from that required for embedded execution: the compact student preserves the teacher's torque mapping over the evaluated sequence and runs in real time on a microcontroller. Together, these results address efficiency at the two stages targeted by ExoSwift—downstream policy development and embedded deployment.

REFERENCES

[1] J. Kim, G. Lee, R. Heimgartner, D. Arumukhom Revi, N. Karavas, D. Nathanson, I. Galiana, A. Eckert-Erdheim, P. Murphy, D. Perry, N. Menard, D. K. Choe, P. Malcolm, and C. J. Walsh, "Reducing the metabolic rate of walking and running with a versatile, portable exosuit," *Science*, vol. 365, no. 6454, pp. 668–672, 2019.

[2] C. Siviy, L. M. Baker, B. T. Quinlivan, F. Porciuncula, K. Swaminathan, L. N. Awad, and C. J. Walsh, "Opportunities and challenges in the development of exoskeletons for locomotor assistance," *Nature Biomedical Engineering*, vol. 7, no. 4, pp. 456–472, 2023.

[3] B. Lim, B. Choi, C. Roh, J. Lee, Y. J. Kim, and Y. Lee, "Ultra-lightweight robotic hip exoskeleton with anti-phase torque symmetry for enhanced walking efficiency," *Scientific Reports*, vol. 15, no. 1, p. 10850, 2025.

[4] Y. Ding, M. Kim, S. Kuindersma, and C. J. Walsh, "Human-in-the-loop optimization of hip assistance with a soft exosuit during walking," *Science Robotics*, vol. 3, no. 15, p. eaar5438, 2018.

[5] P. Slade, M. J. Kochenderfer, S. L. Delp, and S. H. Collins, "Personalizing exoskeleton assistance while walking in the real world," *Nature*, vol. 610, no. 7931, pp. 277–282, 2022.

[6] Y. Chen, S. Miao, G. Chen, J. Ye, C. Fu, B. Liang, S. Song, and X. Li, "Learning to assist different wearers in multitasks: efficient and individualized human-in-the-loop adaptation framework for lower-limb exoskeleton," *IEEE Transactions on Robotics*, vol. 40, pp. 4699–4718, 2024.

[7] S. Choi, C. Ko, and K. Kong, "Walking-speed-adaptive gait phase estimation for wearable robots," *Sensors*, vol. 23, no. 19, p. 8276, 2023.

[8] Y. Qian, Y. Wang, C. Chen, J. Xiong, Y. Leng, H. Yu, and C. Fu, "Predictive locomotion mode recognition and accurate gait phase estimation for hip exoskeleton on various terrains," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6439–6446, 2022.

[9] M. Kim, W.-J. Baek, and J. Park, "A deep reinforcement learning based end-to-end control framework for lower limb exoskeletons with smooth movement transitions," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025, pp. 12 005–12 012.

[10] Y. Yuan, J. Wolf, G. Androwis, and X. Zhou, "Staged multi-agent training (SMAT) for hip exoskeletons: Metabolic and biomechanical validation of a simulation-trained co-adaptive controller," *arXiv preprint arXiv:2608.00715*, 2026.

[11] G. Leem, J. Lee, J. Lee, S. Song, and J. Won, "Exo-Plore: Exploring exoskeleton control space through human-aligned simulation," in *International Conference on Learning Representations (ICLR)*, 2026.

[12] B. Lim, J. Lee, J. Jang, K. Kim, Y. J. Park, K. Seo, and Y. Shim, "Delayed output feedback control for gait assistance with a robotic hip exoskeleton," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 1055–1062, 2019.

[13] S. Luo, M. Jiang, S. Zhang, J. Zhu, S. Yu, I. Dominguez Silva, T. Wang, E. Rouse, B. Zhou, H. Yuk, X. Zhou, and H. Su, "Experiment-free exoskeleton assistance via learning in simulation," *Nature*, vol. 630, no. 8016, pp. 353–359, 2024.

[14] I. Park, C. Song, and I. Kang, "Learning hip exoskeleton control policy via predictive neuromusculoskeletal simulation," *arXiv preprint arXiv:2603.04166*, 2026.

[15] C. Robbins, H. Son, C. K. Tan, C. Wang, R. van Kanten, M. Sartori, G. Durandau, V. Kumar, V. Caggiano, and S. Song, "Myoassist 1.0: An open-source framework for neuromechanical simulation of physical human-device interaction," *bioRxiv*, pp. 2026–08, 2026.

[16] Y. Yuan, J. Wolf, G. Androwis, and X. Zhou, "UniExo: Unified multi-skill policies for musculoskeletal locomotion and co-adaptive exoskeleton control," 2026.

[17] C. Li, C. Wang, B. Ziliotto, M. Simos, J. Kovecses, G. Durandau, and A. Mathis, "Towards embodied AI with MuscleMimic: Unlocking full-body musculoskeletal motor learning at scale," *arXiv preprint arXiv:2603.25544*, 2026.

[18] Y. Zhou, C. Chen, J. Zhang, Y. Liu, R. Zhang, Z. Zhang, and X. Wu, "A simulation-based lower-limb exoskeleton control method," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2025, pp. 1453–1458.

[19] H. Barati, S. Kim, N. T. Xuan, J. Lee, and Y. J. Park, "End-to-end policy learning for hip exoskeleton via reinforcement learning and reflex-based musculoskeletal simulation," *IEEE Robotics and Automation Letters*, vol. 11, no. 7, pp. 8672–8679, 2026.

[20] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen, "OpenSim: Open-source software to create and analyze dynamic simulations of movement," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 11, pp. 1940–1950, 2007.

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

[22] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "DeepMimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 143:1–143:14, 2018.

[23] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of inertial sensors using Allan variance," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 1, pp. 140–149, 2008.

[24] B. C. Abbott, B. Bigland, and J. M. Ritchie, "The physiological cost of negative work," *The Journal of Physiology*, vol. 117, no. 3, pp. 380–390, 1952.

[25] R. Margaria, "Positive and negative work performances and their efficiencies in human locomotion," *Internationale Zeitschrift für Angewandte Physiologie einschließlich Arbeitsphysiologie*, vol. 25, no. 4, pp. 339–351, 1968.

[26] L. I.-K. Lin, "A concordance correlation coefficient to evaluate reproducibility," *Biometrics*, vol. 45, no. 1, pp. 255–268, Mar. 1989.

[27] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *International Conference on Learning Representations (ICLR)*, 2016, arXiv:1506.02438.