Robotics: Science and Systems 2026
Sydney, Australia, July 13–17, 2026

NeuralActuator: Neural Actuation Modeling for Robot Dynamics and External Force Perception

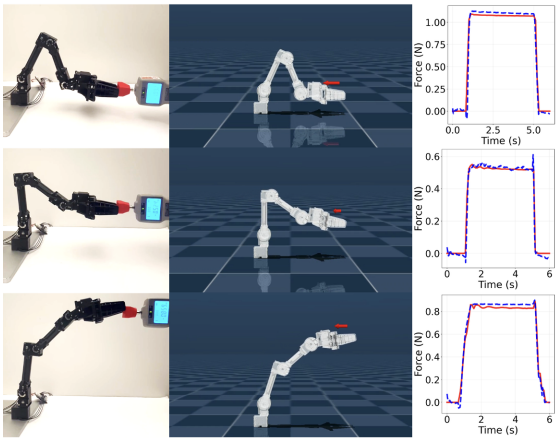
Zhiyang Dou[1], John U. Onyemelukwe[1*], Hangxing Zhang[1*], Heng Zhang[1], Minghao Guo[1], Yunsheng Tian[1], Michal Piotr Lipiec[1], Joshua Jacob[1], Chao Liu[1], Peter Yichen Chen[1], Yuri Ivanov[2,†] and Wojciech Matusik[1]

[1]MIT [2]Amazon Robotics

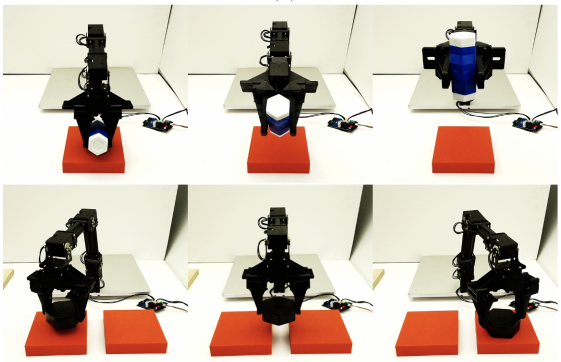
***Abstract*—Differentiable simulators have advanced policy learning and model-based control across diverse robotic tasks. To date, actuator dynamics remain underexplored and are a major source of sim-to-real error, especially on low-cost platforms where the linear current–torque model $\tau = K_t I$ breaks down under commanded-target tracking due to friction, hysteresis, backlash, and thermal effects. Beyond forward dynamics, accurate actuator models also support force perception, which is crucial for jointly modeling force and position control in manipulation tasks. We present NeuralActuator, a neural actuator model that jointly predicts (i) torques to capture the full nonlinear and time-varying current–torque relationship on low-cost servos, (ii) external contact forces as well as force detection gates for sensorless force perception, (iii) motor conditions indicating their operating regime. We introduce a twin-arm teleoperation system that collects motor states alongside ground-truth forces, contributing a dataset named Neural Actuation Dataset (NAD). The torque head is trained through differentiable simulation from pose trajectories without torque labels; the force and motor-condition heads use direct supervision from their respective ground truth. A Transformer-based architecture captures temporal dependencies while enabling efficient real-time inference. We validate NeuralActuator across three platforms—a 5-DoF OpenManipulator-X, a 6-DoF SO-101 from LeRobot, and a 7-DoF Franka Emika Panda with on-board joint torque sensors—spanning three actuator families and the $500–$30k+ cost spectrum, and show that it enables accurate dynamics modeling, sensorless force estimation, motor condition estimation, and improved behavior-cloning control when used as a pretrained module. Project Page: https://frank-zy-dou.github.io/projects/NeuralActuator/index.html**

I. Introduction

Differentiable simulators [23, 35, 14, 21, 2, 44] enable efficient gradient-based optimization for robot control, with substantial advances in modeling rigid-body dynamics and recent extensions to soft bodies and fluids. However, a crucial and often underexplored component remains: *the actuators themselves*. For decades, control frameworks have assumed joint torque scales linearly with current, e.g., $\tau = K_t I$. While this could hold for high-end industrial motors, it degrades substantially on cost-effective, servo-driven platforms [5, 39, 34, 40]. These platforms employ mechanically commutated actuators (e.g., coreless brushed DC servo modules) that prioritize hardware simplicity but introduce significant non-idealities: gearbox friction, hysteresis, backlash, current saturation, and thermal drift distort the actual joint torque; see visualization in Sec. IV-J. These unmeasured, time-varying effects—with inaccessible internal signals—resist analytical modeling, resulting in poor torque tracking, unstable impedance behavior, and degraded sim-to-real transfer.



(a)



(b)

Fig. 1: **(a) Push–force gauge validation at three end-effector heights.** We show NeuralActuator pushing a force gauge under *low*, *middle*, and *high* postures (top to bottom). Left: real-robot execution; middle: corresponding simulated rollout; right: force profiles. The force Mean Absolute Error (MAE) is 0.037 N, 0.015 N, and 0.028 N, respectively. The - - - shows the predicted external force, and the —— shows the force-gauge ground truth. **(b) Estimated forces for downstream tasks.** NeuralActuator provides estimated forces that facilitate controller training in Sim-to-Real for object lifting and holding (top) and pick-and-place (bottom) tasks.

Beyond accurate modeling, a reliable actuator model unlocks *force perception without dedicated sensors*. Recent studies demonstrate torque-based sensing for e-skin–like contact perception [25], proprioceptive estimation in legged robots [15], and force control in manipulation [29, 18, 10, 51]. Yet these approaches rely on high-end actuators with reliable

*Research Assistant at MIT CDFG, equal contribution.
†The work of this author does not relate to their position at Amazon.

torque feedback; on low-cost platforms, the corrupted current-to-torque mapping undermines such force inference. Since the medium of interaction between a manipulator and its environment is ultimately *force*, accurate contact-force prediction makes many tasks tractable without precise geometric modeling of arbitrary or deformable objects.

We introduce **NeuralActuator**, a data-driven actuation modeling method for low-cost robot platforms that captures actuator behavior and its coupling with robot dynamics. NeuralActuator integrates naturally with a differentiable simulator, enabling end-to-end training via gradient-based supervision from real-robot rollouts. NeuralActuator learns a history-dependent mapping from commanded poses, motor currents, and exogenous operating conditions (e.g., supply voltage, temperature) to joint torques, external forces, and motor health indicators. Prior work that trains actuator models typically relies on torque labels derived from torque sensors [41, 24], but such supervision is unreliable on low-cost actuators without dedicated torque sensing. Instead, we supervise using pose trajectories: forward integration through our differentiable dynamics maps predicted torques to future configurations, making pose an effective surrogate signal. To effectively capture nonlinear, time-varying actuator effects, NeuralActuator adopts a Transformer architecture that processes state-and-command sequences through multi-head self-attention, capturing long-horizon temporal dependencies and inter-joint coupling. It outputs joint-specific torque predictions, external forces, and motor condition estimates in a multi-task manner.

To train and benchmark NeuralActuator, we introduce a real-robot dataset capturing actuation, motion, and force signals. We build a twin-arm leader–follower teleoperation system where an operator kinesthetically drives a leader arm to generate diverse trajectories, while the follower executes joint-space commands under closed-loop control. For each trajectory, we log (i) robot states (joint positions, velocities, commanded targets), (ii) actuator telemetry (motor currents, temperature, supply voltage), and (iii) external forces from force sensors. The dataset spans free-space motions and contact-rich manipulation (pushing, pulling, lifting, known-weight loads), covering actuator effects induced by friction, load changes, and intermittent contacts. We release it as the **Neural Actuation Dataset (NAD)**, providing ground-truth data for model training and benchmarking.

We conduct several experiments across three platforms—a cost-effective, servo-driven 5-DoF OpenManipulator-X (four revolute joints and a 1-DoF gripper) [1], a 6-DoF SO-101 low-cost arm built on the LeRobot stack [5], and a 7-DoF industrial arm, i.e., Franka Emika Panda robot, with on-board joint torque sensors (Sec. IV-I)—spanning three actuator families and the $500–$30k+ cost spectrum, to evaluate modeling fidelity, online adaptation, throughput, and sim-to-real transfer. On held-out rollouts, NeuralActuator achieves low rollout error (Sec. IV-A), accurate external force estimation (Sec. IV-B) as well as motor condition estimation (Sec. IV-D). The simulator attains high runtime speed and throughput with a modest memory footprint, making it practical for both training and on-robot deployment (Sec. IV-E). We also showcase rapid online adaptation from only few new trajectories (Appendix I). For sim-to-real control, NeuralActuator enables efficient imitation learning for object manipulation. We show that learning actuator behavior together with external force perception improves task success rates (Sec. IV-F). Finally, we demonstrate that combining differentiable simulation with multiple rendering modalities enables visual supervision for NeuralActuator training (Sec. IV-G). We summarize our contributions below:

- A data-driven, differentiable actuator model named **NeuralActuator** that captures actuator behavior and its coupling with robot dynamics via neural torque prediction, with additional outputs for external force prediction and contact detection—enabling force perception without dedicated sensors—and a torque-supervision-free training scheme that leverages differentiable physics to avoid unreliable current–torque labels on low-cost hardware.
- A twin-arm teleoperation system for efficient data collection on low-cost arms, time-synchronizing robot proprioception, commanded targets, and actuator-side telemetry.
- **Neural Actuation Dataset (NAD)**: a real-robot dataset with time-aligned robot states, actuator telemetry, and end-effector force measurements from interactions measured by external force sensors, enabling training and benchmarking for low-cost actuator modeling.
- Comprehensive experiments demonstrate that NeuralActuator enables accurate actuator dynamics modeling and sensorless force as well as motor-condition estimation, and improves performance on downstream manipulation tasks, while remaining computationally efficient.

II. Related Work

A. Learning Neural-Augmented Simulation

To close the sim-to-real gap, prior works augment analytical simulators with learned residual models: the approach in [1] augments analytical simulators with a stochastic RNN that captures residual dynamics to match the simulation with reality. In [26], an RNN is trained to learn the mismatch in joint positions and velocities between simulated and real robot trajectories for refining the simulator output. TossingBot [50] further combines an analytic ballistic prior with a neural residual learned from visual features. NeuralSim [19] embeds neural modules in differentiable simulators to capture unmodeled dynamics and jointly optimizes them with physical parameters on real data; at the component level, Serifi et al. [42] uses a Transformer to predict residual state corrections for actuators and rigid bodies from simulated states and constraint forces. Recently, [48] presents NeRD, which learns robot-specific dynamics models for predicting future states for articulated rigid bodies under contact constraints. [24] trains a network on the physical system via self-supervision and uses it in the simulation loop to model each joint of

[1] https://emanual.robotis.com/docs/en/platform/openmanipulator_x/overview/

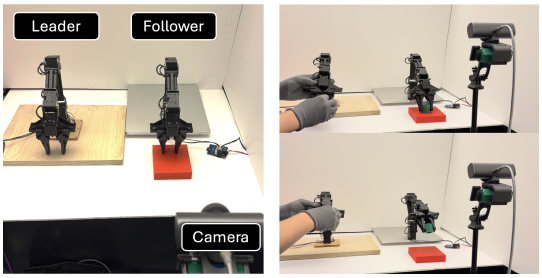
ANYmal; the network maps state histories and position targets to joint torques for simulation. [41] introduces a system identification method that combines a physics model with a neural network to learn residual dynamics, enabling domain-consistent dynamics modeling. In their setting, joint torques can be accurately estimated from motor-current measurements. However, these works [24, 41] rely on ground-truth torque labels from dedicated sensors for supervision, which are often unavailable on low-cost platforms. In contrast, NeuralActuator is *torque-label-free*: it is supervised through differentiable simulation using pose trajectories, without assuming reliable current–torque calibration. Concurrently, [13] learns residual corrective torques via RL from tracking errors; NeuralActuator instead directly predicts forward torques (raw current readings are too noisy on low-cost platforms; see Fig. 9) and additionally estimates external contact forces, trained end-to-end via differentiable simulation with pose-only supervision.

a) Learning for Actuator: Existing works mostly target electronically commutated drives (e.g., PMSMs and BLDCs), learning electromagnetic torque from electrical inputs [49, 32, 46, 43]. These approaches assume high-performance, electronically commutated servos. Previous works also cover behaviors of hydraulics [27] and twisted-string actuators [30]. Unlike these efforts, NeuralActuator targets low-cost *coreless DC* servo actuators, where friction, backlash, saturation, and thermal drift are more pronounced and can corrupt the effective current–torque relationship.

B. Force Estimation and Proprioceptive Sensing

a) Torque- and effort-based contact inference: A long-standing line of work reconstructs external joint torques from model discrepancies and uses them for collision detection and safe reaction [11, 12, 16]. Building on reliable effort feedback, recent systems further elevate this idea to richer "intrinsic" contact perception (e.g., e-skin–like touch without tactile skins) [25], whole-robot touch sensing without dedicated tactile sensors [15], and learning-based controllers that couple force and motion in manipulation and loco-manipulation [29, 51]. These approaches highlight the growing role of actuation as an implicit sensing channel, but they often rely on high-quality torque measurements or well-calibrated current-to-torque mappings.

b) External force estimation on robot arms: For manipulators, complementary literature estimates external forces and contacts by explicitly modeling the robot dynamics and attributing residual terms to interaction wrenches. Representative examples include virtual force sensing and contact localization along the robot structure [36, 37], as well as observer and filtering formulations that improve robustness under friction and model uncertainty [22, 33, 17, 31]. Recently, learning has also been used to directly map joint-level proprioception and measured joint torques from sensors to end-effector wrenches in static or quasi-static regimes [38]. Chen et al. [8] leverage differentiable simulation to infer external object properties solely from proprioceptive signals. Despite this progress, most proprioceptive force-estimation methods assume accurate torque feedback, while the setting of low-cost, servo-driven arms remains largely unexplored. A summary can be found in Tab. A15 in Appendix.



(a) (b)

Fig. 2: **Neural Actuation Dataset (NAD) data collection.** (a) Overview of the twin-arm leader–follower system with an external camera. (b) An operator kinesthetically drives the leader to generate diverse motions, while the follower mirrors the motion and records synchronized actuator currents, robot states, and end-effector forces when interacting with a weight.

III. Method

A. Problem Formulation

Consider a robotic manipulator with n joints, where each joint is driven by a servo motor. The relationship between commanded pose, current, and output torque exhibits dependencies on multiple factors:

$$\tau_j^{\text{real}}(t) = f_\tau(q_j^{\text{cmd}}(t), i_j(t), q_j(t), \dot{q}_j(t), T_j(t), \mathcal{H}_t) \quad (1)$$

where $q_j^{\text{cmd}}(t)$ is the commanded (goal) position, $q_j(t)$ and $\dot{q}_j(t)$ are joint position and velocity, $T_j(t)$ represents thermal state, and $\mathcal{H}_t = \{(q_j^{\text{cmd}}(\tau), i_j(\tau), q_j(\tau), \dot{q}_j(\tau))\}_{\tau=t-w}^{t}$ denotes the historical window of states over time horizon w.

Beyond torque prediction, we estimate external contact forces and motor conditions from the same input features:

$$\mathbf{f}^{\text{ext}}(t) = f_{\text{force}}\Big(q^{\text{cmd}}(t), i(t), q(t), \dot{q}(t), T(t), \mathcal{H}_t\Big) \in \mathbb{R}^3 \quad (2)$$

$$c(t) = f_{\text{cond}}\Big(q^{\text{cmd}}(t), i(t), q(t), \dot{q}(t), T(t), \mathcal{H}_t\Big) \in [0, 1] \quad (3)$$

where $\mathbf{f}^{\text{ext}}(t)$ is the predicted end-effector contact force and $c(t)$ is a motor health score (probability of normal operation, with $c = 1$ for healthy and $c = 0$ for degraded motors). A physical interpretation of these outputs in terms of the rigid-body equations of motion is given in Appendix B.

B. Neural Actuation Dataset (NAD)

To learn a model for Actuation Modeling, we first collect paired motor-state and force data using a twin-arm teleoperation system built on two identical OpenManipulator-X robots [40]. The system consists of a *leader* arm kinesthetically operated by a human demonstrator and a *follower* arm that mirrors the leader's motion while interacting with the environment (Fig. 2).

Note that in this system, the *commanded target joint positions* sent to the follower's actuators are set to the leader's

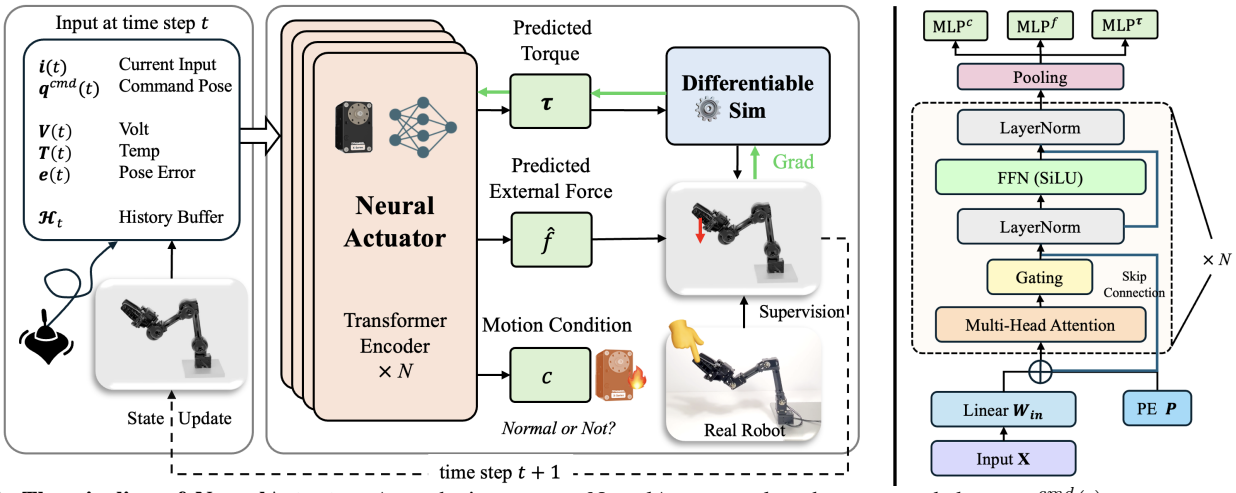


Fig. 3: **The pipeline of NeuralActuator.** At each time step t, NeuralActuator takes the commanded pose $q^{cmd}(t)$, motor current $i(t)$, actuator-side telemetry, and tracking feedback $e(t)$, together with a history buffer $\mathcal{H}_t$ that summarizes recent robot states (e.g., joint poses and velocities). A Transformer encoder ($\times N$ blocks) maps these inputs to three outputs: (i) the *directly predicted* joint torque τ, (ii) a predicted external force $\hat{f}$, and (iii) a motor-condition signal c indicating the operating regime (e.g., normal vs. anomalous). The predicted τ and $\hat{f}$ are applied in a differentiable simulator (*Differentiable Simulation*) to advance the system to time step $t+1$ and update the history buffer. Gradients through *Diff Sim* enable end-to-end training by supervising simulated state transitions against real-robot rollouts. **Right:** Transformer block details with input projection W_{in}, positional encoding, multi-head attention, a gated skip connection, SiLU feed-forward network, pooling, and lightweight MLP heads for τ, $\hat{f}$, and c.

instantaneous joint states. Both arms are equipped with Dynamixel XM430-W350 servos, which provide synchronized actuator-level measurements, including:

- Joint positions $q_j(t)$ and velocities $\dot{q}_j(t)$
- Motor currents $i_j(t)$, PWM signals, and bus voltages
- Coil temperatures $T_j(t)$

For a subset of interaction trajectories, the follower arm is further instrumented with a wrist-mounted 3-axis force sensor, which provides ground-truth end-effector contact force measurements $\mathbf{f}^{\text{gt}}(t) \in \mathbb{R}^3$ during execution.

All data streams are temporally synchronized and logged per trajectory. For each trajectory of length T, we record time-aligned tuples containing: (i) robot states (e.g., joint positions, velocities, and commanded targets), (ii) motor currents and actuator-side telemetry, and (iii) end-effector external forces, which provides supervision for training a neural actuation model. To elicit diverse actuation and interaction regimes, we collect three subsets:

Free motion (no external force). This subset characterizes intrinsic actuator dynamics and provides force-free baselines. We record a diverse set of trajectories (119,933 frames, avg. 20.49 s), including: (a) Clockwise and counterclockwise circular end-effector motions; (b) individual joint sweeps covering approximately 90% of feasible ranges; and (c) whole-arm primitives such as *lean and extend*. These trajectories contain no intentional external contact.

Force-labeled interactions. This subset provides explicit force supervision via two modalities. *(a) Known-weight loads.* The robot manipulates objects of known mass ($m \in \{200, 300, 400, 500\}$ g). During *loaded/grasped* intervals, the static payload yields a base-frame gravity vector $\mathbf{f}^{\text{gt}}_{\text{grav}} = m\,\mathbf{g}_{\text{base}} \in \mathbb{R}^3$ (with $\mathbf{g}_{\text{base}} = [0, 0, -g]^\top$), which we use as the force-magnitude and vertical-component label during those intervals only; dynamic terms and gripping friction

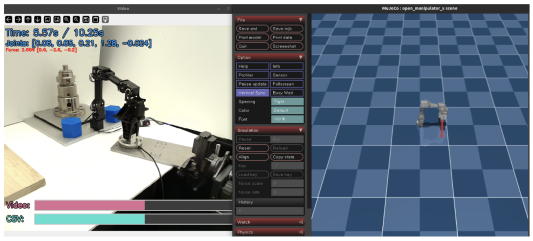
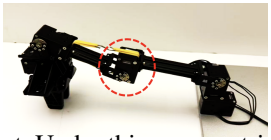


Fig. 4: **Data verification in NAD.** We time-synchronize the real-robot video (left) with the logged trajectory and verify alignment in a GUI (right), where the red arrow visualizes the estimated external force.

are not labeled here. Tasks include *go up and stay still* (persistent load) and *pick and place* (intermittent load during grasping), totaling 88,611 frames with an average duration of 21.63 s. *(b) Axis-aligned sensor interactions.* Operators apply controlled push and pull forces along the force sensor's principal axes ($\pm X, \pm Y, \pm Z$). For each force direction, we also record matched no-interaction trajectories that follow the same motion patterns but are executed without physical contact. The measured wrench is transformed into the robot base coordinate to capture full 3D force components arising from both dominant-axis interactions and kinematic coupling (74,020 frames, avg. 10.50 s).

We further collect additional data for the cross-platform evaluation in Sec. IV-I, using both SO-101 logged via the LeRobot stack [5] and a 7-DoF Franka Emika Panda industrial arm, whose on-board torque sensors provide ground-truth joint torques for direct validation.



Manipulation under degraded motor conditions. In this task, we mechanically constrain Joint 3 using rubber bands, reducing its effective range and torque capacity; see the inset. Under this asymmetric constraint, we collect *pick and place* trajectories with and with-

TABLE I: **Dataset composition summary.** See Appendix A for a detailed task-level breakdown.

Component	Description	Duration (min)
Free motion	No external force	~ 34.15
Force labeled	Known weights or force	~ 46.24
Motor condition	Degraded actuation	~ 14.13
Total		~ 94.52

out a 200 g payload (49,898 frames, avg. 21.20 s), enabling disentanglement of contact-induced and motor-induced force deviations.

Our dataset is split into training, validation, and test sets with a ratio of 8:1:1, which supports (i) quantitative evaluation against known ground-truth forces, (ii) qualitative analysis of force prediction during contact-rich manipulation, and (iii) robust modeling of contact-free actuation dynamics for improved generalization. All the data in NAD are calibrated and human-verified as shown in Fig. 4. See more details in Appendix A.

C. NeuralActuator

To capture temporal dependencies and nonlinear actuator dynamics, we employ a Transformer encoder architecture that processes historical state sequences and generates torque corrections. Given a sequence length L and hidden dimension d, our model consists of:

In NeuralActuator, the input sequence $\mathbf{X} \in \mathbb{R}^{L \times n_f}$ stacks per-timestep feature vectors $\mathbf{x}_t \in \mathbb{R}^{n_f}$ over a history window of $L = H + 1$ timesteps. Each $\mathbf{x}_t$ aggregates three families of signals—(i) commanded targets $(q^{\text{cmd}}, a^{\text{cmd}})$, (ii) proprioception $(q, \dot{q}, a)$, and (iii) actuator telemetry (i, V, T)—together with tracking-error features $(e = q^{\text{cmd}} - q,\ e_a = a^{\text{cmd}} - a)$. All features are normalized before being fed to the network. Note that for grippers, we derive and validate the mapping from Motor 5 rotation to gripper aperture for simulation and training; see Appendix G.

We project the input sequence to the hidden dimension via $\mathbf{H}_{\text{enc}}^{(0)} = \mathbf{X}\mathbf{W}_{\text{in}}$, where $\mathbf{W}_{\text{in}} \in \mathbb{R}^{n_f \times d}$ is the input projection matrix.

We stack N_{enc} Transformer encoder layers to extract contextual representations from the input sequence:

$$\mathbf{Z}_{\text{enc}}^{(l)} = \text{LayerNorm}\Big(\mathbf{H}_{\text{enc}}^{(l-1)} + \gamma_a^{(l)} \cdot \text{MHA}\Big(\mathbf{H}_{\text{enc}}^{(l-1)}\Big)\Big), \quad (4)$$

$$\mathbf{H}_{\text{enc}}^{(l)} = \text{LayerNorm}\Big(\mathbf{Z}_{\text{enc}}^{(l)} + \gamma_f^{(l)} \cdot \text{FFN}\Big(\mathbf{Z}_{\text{enc}}^{(l)}\Big)\Big), \quad (5)$$

where MHA denotes multi-head self-attention, FFN is a feed-forward network with SiLU activation, and $\gamma_a^{(l)}, \gamma_f^{(l)} \in \mathbb{R}$ are learned scalar gates for the attention and FFN skip connections. After the encoder layers, we aggregate the sequence representation via temporal pooling, $\mathbf{h}_{\text{pool}} = \text{Pool}(\mathbf{H}_{\text{enc}}^{(N_{\text{enc}})}) \in \mathbb{R}^d$, which is passed to the output heads. The network produces three types of outputs from the pooled representation $\mathbf{h}_{\text{pool}}$:

1. Torque Prediction. NeuralActuator achieves the torque prediction via

$$\boldsymbol{\tau}^{\text{pred}}(t) = g_\theta(\mathcal{H}_t), \quad (6)$$

where $\boldsymbol{\tau}^{\text{pred}}(t) \in \mathbb{R}^n$ denotes the predicted joint torques and $\mathcal{H}_t = \{\mathbf{x}_{t-w}, \ldots, \mathbf{x}_t\}$ is a history window of proprioception, commands, and actuator telemetry. This design is motivated by the fact that the effective current–torque relationship on low-cost servo-driven arms can be highly nonlinear and time-varying during closed-loop tracking; See visualization in Sec. IV-J. Direct torque prediction lets the network learn the full nonlinearity without relying on a fixed linear prior.

A shared MLP predicts joint torques for all joints: $\boldsymbol{\tau}^{\text{pred}} = \text{MLP}_\tau(\mathbf{h}_{\text{pool}}) \in \mathbb{R}^n$, where MLP_τ consists of two linear layers with SiLU activation.

2. External Force Prediction. We predict end-effector contact forces using a two-stage approach that decouples force magnitude estimation from contact detection. First, a shared MLP predicts the raw 3D contact force: $\hat{\mathbf{f}}_{\text{raw}} = \text{MLP}^{\text{force}}(\mathbf{h}_{\text{pool}}) \in \mathbb{R}^3$, which estimates force components from motor states alone, without dedicated force sensors. To suppress spurious force predictions during non-contact phases, we introduce a *contact gate* $g \in [0, 1]$ predicted by a separate head: $g = \sigma(\text{MLP}^{\text{gate}}(\mathbf{h}_{\text{pool}}))$, where $\sigma(\cdot)$ denotes the sigmoid function. The gate represents the probability of physical contact: $g \approx 1$ indicates contact, while $g \approx 0$ indicates free motion. The final force output is computed as: $\hat{\mathbf{f}} = g \cdot \hat{\mathbf{f}}_{\text{raw}}$. During training, the gate supervision is derived from ground-truth force measurements $g_{\text{gt}} = \mathbb{I}\left[\|\mathbf{f}_{\text{gt}}\|_2 > \epsilon\right]$, where $\epsilon = 0.01\,\text{N}$ serves as a noise threshold.

3. Motor Condition Estimation. A separate head predicts a motor condition score $c \in [0, 1]$: $c = \sigma(\text{MLP}^{\text{cond}}(\mathbf{h}_{\text{pool}}))$, where $c = 1$ indicates normal motor operation and $c = 0$ indicates degraded or damaged motor state. This enables condition monitoring by detecting anomalous patterns from current-torque discrepancies and thermal signatures.

Differentiable Simulation Integration. NeuralActuator is integrated with differentiable physics simulators. We embed our model within the simulation loop to enable gradient-based optimization through the entire system dynamics.

Forward Dynamics. At each simulation timestep t, the simulator computes: (1) Predicted torque: $\boldsymbol{\tau}_t^{\text{pred}} = g_\theta(\mathcal{H}_t)$ and (2) State evolution: $\mathbf{s}_{t+1} = \text{DiffSim}(\mathbf{s}_t, \boldsymbol{\tau}_t^{\text{pred}}, \hat{\mathbf{f}}_t^{\text{ext}})$, where $\mathbf{s}_t = [q_t, \dot{q}_t]$ represents the joint state and q_t^{cmd} is the commanded position.

Gradient Computation. Through automatic differentiation, we compute gradients of a trajectory loss $\mathcal{L}$ with respect to network parameters:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial s_t} \frac{\partial s_t}{\partial \tau_t^{\text{pred}}} \frac{\partial \tau_t^{\text{pred}}}{\partial \theta}. \quad (7)$$

This enables end-to-end learning where the network parameters are optimized to minimize discrepancies between simulated and real trajectories.

Model Training. Since ground-truth torque values cannot be directly measured on low-cost platforms, the torque head is supervised through pose trajectories: forward integration maps predicted torques to future configurations, making pose measurements effective surrogate targets. The force, gate, and motor-condition heads use direct supervision from their respective ground truth. We minimize discrepancies between

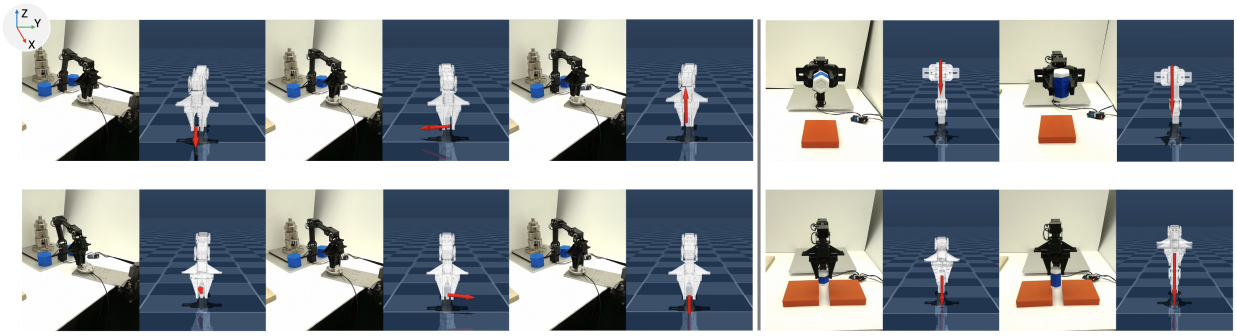


Fig. 5: **Robot arm simulation with external force estimation on the test set.** (a) NeuralActuator estimates end-effector external forces along six directions ($\pm X, \pm Y, \pm Z$); red arrows visualize the estimated force vectors in simulation (paired with corresponding real-robot executions). (b) Force estimation during force-aware manipulation: lifting and holding 200 g and 300 g payloads (top) and pick-and-place with 200 g and 300 g payloads (bottom). Please refer to our video for more results.

simulated and real trajectories through five loss components. *Joint Angle Loss:* To handle angular wraparound, we define $\text{wrap}_\pi(x) = ((x+\pi) \bmod 2\pi) - \pi$ and compute $\mathcal{L}_{\text{joint}} = \frac{1}{|\mathcal{T}|}\sum_t \|\text{wrap}_\pi(\phi_t^{\text{sim}} - \phi_t^{\text{real}})\|_1$. *Gripper Loss:* For the prismatic gripper, $\mathcal{L}_{\text{grip}} = \frac{1}{|\mathcal{T}|}\sum_t |q_{\text{grip},t}^{\text{sim}} - q_{\text{grip},t}^{\text{real}}|$. *Force Loss:* When ground-truth forces are available, $\mathcal{L}_{\text{force}} = \frac{1}{|\mathcal{T}|}\sum_t \|\mathbf{f}_t^{\text{pred}} - \mathbf{f}_t^{\text{gt}}\|_1$. *Gate Loss:* The contact gate is supervised with binary cross-entropy against $g_{\text{gt}} = \mathbb{I}[\|\mathbf{f}_{\text{gt}}\|_2 > \epsilon]$. *Condition Loss:* Motor condition is supervised with binary cross-entropy: $\mathcal{L}_{\text{cond}} = -\frac{1}{|\mathcal{T}|}\sum_t [s_t \log c_t + (1-s_t)\log(1-c_t)]$, where $s_t = 1$ for normal motors and $s_t = 0$ for degraded (restricted) motors. The total loss is: $\mathcal{L} = \lambda_{\text{joint}}\mathcal{L}_{\text{joint}} + \lambda_{\text{grip}}\mathcal{L}_{\text{grip}} + \lambda_{\text{force}}\mathcal{L}_{\text{force}} + \lambda_{\text{gate}}\mathcal{L}_{\text{gate}} + \lambda_{\text{cond}}\mathcal{L}_{\text{cond}}$, where $\mathcal{T}$ denotes timesteps with available ground-truth. We apply focal weighting for force samples to address contact/free-motion imbalance, and train with Adam [28] for 10000 epochs. Unless otherwise specified, experiments are conducted on a standard workstation with an Intel Core Ultra 7 265K processor (20 cores, 5.0 GHz), 32 GB of memory, and an NVIDIA GeForce RTX 5080 GPU with 16 GB of VRAM; the no-load benchmark in Appendix C uses an RTX 4090, and the hand-eye calibration measurements in Appendix D use an A6000. Network training and inference were implemented in JAX JIT [4] with GPU acceleration, and differentiable simulation for rigid body dynamics was based on MuJoCo MJX [44]. More details are in our Appendix.

IV. Experiments

A. Rollout Accuracy

We evaluate the fidelity of our differentiable simulator by comparing predicted trajectories against ground truth on a held-out test dataset. For each joint j at horizon T, we compute the mean absolute error (MAE) across N test trajectories: $\text{MAE}_j(T) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{T}\sum_{t=1}^{T}|q_{j,t}^{(n)} - \hat{q}_{j,t}^{(n)}|$, where $q_{j,t}^{(n)}$ and $\hat{q}_{j,t}^{(n)}$ denote the ground truth and predicted positions, respectively. Tab. III presents the simulation accuracy across different time horizons. Our model achieves relatively high short-term accuracy for revolute joints and prismatic joints. The error growth remains bounded even at longer horizons—a critical property for model-predictive control. We believe this horizon is sufficient for most online planning applications, where replanning typically occurs at higher frequencies. The consistent error growth pattern across joints validates our physics-in-the-loop learning approach, which captures actuator-specific, nonlinear effects while remaining compatible with differentiable rigid-body dynamics.

TABLE III: **Simulation accuracy on the test set.** J1–J4: joint-angle MAE (deg). Grip: gripper-aperture MAE (mm). @600 steps correspond to 10 seconds.

Task	@100 steps					@300 steps					@600 steps				
	J1	J2	J3	J4	Grip	J1	J2	J3	J4	Grip	J1	J2	J3	J4	Grip
backward_forward	2.4	4.3	4.4	4.2	0	2.3	5.9	3.9	4.2	0	3.1	4.8	4.9	4.5	0
circular_ccw	1.7	5.2	2.2	2.6	0	1.5	3.1	2.0	3.4	0	2.5	2.3	2.1	2.8	0
circular_cw	3.6	3.2	1.4	2.2	0	3.1	1.9	1.2	2.6	0	2.6	1.7	2.0	2.8	0
go_up_stay_still	2.5	1.8	2.3	1.2	0	3.1	2.1	3.1	2.5	0	3.1	1.5	3.5	2.7	0
joint_sweep_1	2.3	2.5	1.8	1.4	0	1.9	1.0	2.6	1.8	0	2.9	1.9	2.0	1.6	0
joint_sweep_2	2.1	4.3	4.6	1.2	0	1.8	6.7	8.4	1.1	0	2.4	4.6	5.2	1.7	0
joint_sweep_3	3.7	4.5	3.1	1.9	0	3.4	2.6	2.8	2.8	0	3.1	2.6	2.5	3.4	0
joint_sweep_4	2.6	4.8	2.5	3.4	0	2.9	3.3	2.4	3.2	0	2.4	2.9	2.2	4.6	0
joint_sweep_5	1.7	3.0	1.7	3.1	0.4	2.6	3.9	2.3	2.7	0.5	5.5	3.3	5.1	3.0	0.7
pick_place_empty	2.6	2.5	2.2	3.5	1.1	1.9	3.6	2.7	3.2	1.1	3.1	2.7	2.0	3.9	1.0
Average	2.5	3.6	2.6	2.5	0.2	2.5	3.4	3.1	2.8	0.2	3.1	2.8	3.2	3.1	0.2

TABLE IV: **Simulation and force prediction accuracy on the force-sensor test set.** J1–J4: joint-angle MAE (deg). Grip: gripper-aperture MAE (mm). F: force MAE (N).

Task	@100 steps						@300 steps						@500 steps					
	J1	J2	J3	J4	Grip	F	J1	J2	J3	J4	Grip	F	J1	J2	J3	J4	Grip	F
With External Force Contact																		
force_X+	0.9	2.3	0.9	1.6	1.0	0.36	1.1	1.8	2.4	1.3	1.0	0.46	0.9	1.9	2.7	1.7	1.0	0.50
force_X−	1.4	1.1	0.8	1.1	1.0	0.39	1.6	2.3	1.3	2.3	1.0	0.42	2.0	2.9	1.5	3.0	1.0	0.44
force_Y+	2.3	5.7	2.8	0.6	0.1	0.36	2.1	4.5	2.6	0.9	0.2	0.38	2.7	4.8	2.1	0.9	0.3	0.44
force_Y−	1.4	1.6	3.1	0.5	0.1	0.38	1.5	3.0	3.1	1.3	0.1	0.36	1.4	3.6	3.1	1.5	0.2	0.38
force_Z+	3.3	6.2	3.3	0.8	0.1	0.39	2.3	5.6	2.5	1.9	0.2	0.43	2.5	4.4	2.2	2.3	0.3	0.46
force_Z−	1.0	5.9	0.6	0.6	1.0	0.36	2.9	6.6	2.8	2.5	1.0	0.64	3.8	6.4	3.5	2.0	1.0	0.57
Reference (No External Force)																		
force_X+	0.6	2.0	1.5	1.2	1.0	0.01	0.8	1.8	1.0	1.1	1.0	0.02	1.0	1.6	1.1	1.1	1.0	0.01
force_X−	2.0	2.5	0.5	1.1	1.0	0.00	1.4	2.3	0.5	1.0	1.0	0.00	1.5	2.2	0.8	1.3	1.0	0.00
force_Y+	2.0	1.0	2.0	0.6	0.1	0.00	2.1	3.3	2.1	1.1	0.2	0.00	1.9	3.7	1.7	1.1	0.4	0.00
force_Y−	1.0	1.8	1.2	1.3	0.1	0.00	1.6	4.5	2.0	2.2	0.1	0.00	1.2	4.0	2.4	1.6	0.3	0.00
force_Z+	1.2	1.7	2.1	0.5	0.1	0.00	1.6	3.8	2.2	1.2	0.2	0.00	1.5	2.9	2.0	1.3	0.3	0.00
force_Z−	0.6	2.3	0.9	0.9	1.0	0.02	0.7	1.6	1.1	1.1	1.0	0.01	0.9	1.3	1.0	1.1	1.0	0.01
Avg	1.48	2.84	1.64	0.90	0.55	0.19	1.64	3.43	1.97	1.49	0.58	0.23	1.78	3.31	2.01	1.58	0.65	0.23

Sweeping the rollout horizon $H \in \{64, 128, 256, 320, 500\}$ (Tab. II), simulator-side torque gradients remain $O(10^{-2})$ with no collapse and the parameter-gradient direction stays nearly fixed (cosine alignment ≥ 0.96 relative to H=128); $\|\nabla_\theta L\|$ plateaus around H=256–320, motivating the curriculum used in training.

TABLE II: **Gradient behavior across rollout horizons H.**

H	torque-grad	$\|\nabla_\theta L\|$	cos
64	2.73e-2	8.77	0.96
128	2.59e-2	17.33	1.00
256	1.72e-2	22.35	0.99
320	1.45e-2	22.91	0.99
500	9.37e-3	20.29	0.98

B. Force Estimation Accuracy

We evaluate force prediction on (i) an *external force sensing* benchmark using a calibrated 6-DoF force-torque sensor, and

TABLE V: **Simulation and force prediction accuracy on the weight-based test set.** J1–J4: joint-angle MAE (deg). Grip: gripper-aperture MAE (mm). F: force MAE (N).

Task	Weight	@100 steps						@300 steps						@600 steps					
		J1	J2	J3	J4	Grip	F	J1	J2	J3	J4	Grip	F	J1	J2	J3	J4	Grip	F
go up and stay	200g	1.3	5.0	2.6	2.4	0.1	0.12	3.4	3.9	4.2	2.3	0.1	0.10	3.1	3.3	4.3	2.9	0.1	0.16
	300g	2.6	5.3	5.7	3.7	0.1	0.20	1.8	3.4	5.7	5.4	0.0	0.17	3.5	2.5	3.4	4.6	0.0	0.20
	400g	2.2	4.1	1.4	5.0	0.1	0.07	2.9	2.1	2.1	6.1	0.1	0.12	2.0	1.4	3.0	7.1	0.2	0.11
pick and place	200g	0.7	3.1	4.3	3.4	1.1	0.24	2.7	3.9	4.0	4.0	1.1	0.08	2.9	5.1	2.9	3.0	1.0	0.11
	300g	0.5	4.5	3.7	0.9	0.3	0.00	1.2	5.1	4.8	1.5	0.2	0.00	2.7	7.8	4.9	3.2	0.3	0.09
	400g	0.8	2.1	2.8	2.9	1.1	0.19	2.8	1.4	3.4	3.6	1.1	0.06	2.4	2.0	3.8	3.3	1.0	0.03
	500g	2.7	1.8	1.7	3.3	1.1	0.05	4.8	2.8	1.7	2.3	1.1	0.02	4.2	6.3	2.3	2.3	0.9	0.04
Avg		1.54	3.70	3.17	3.09	0.56	0.12	2.80	3.23	3.70	3.60	0.53	0.08	2.97	4.06	3.51	3.77	0.50	0.11

(ii) a *known-weight* benchmark where ground-truth forces are defined by gravitational loads (200 g–500 g). See visualization in Fig. 5.

External Force Sensing. Tab. IV reports 3D external force prediction across six directions ($\pm X, \pm Y, \pm Z$), with forces up to 9 N applied at the end-effector. On contact trajectories, the force MAE is 0.36–0.39 N at 100 steps and 0.38–0.57 N at 500 steps (4–6% relative error). On reference trajectories without contact, NeuralActuator predicts near-zero forces (0.00–0.02 N), reliably distinguishing contact from no-contact states.We found payloads at or below $\sim$50 g ($\sim$0.5 N) sit close to the noise floor of this low-cost platform. Reliable detection in that regime is left to future work.

Weight Benchmark. Tab. V summarizes force prediction accuracy across weight loads and prediction horizons. NeuralActuator achieves an average force MAE of 0.08–0.12 N across tasks and horizons, with single-task errors below 0.24 N. The go-up-and-stay task exhibits relatively larger per-row errors (up to 0.20 N) under sustained static loading.

Novel Contact Geometries. Because the force head operates at the joint-torque level, it is less tied to explicit object geometry than vision-based force inference. To probe this, we evaluate the pretrained model on two unseen objects (inset) with shapes and surface frictions absent from training (a 261 g and a 226 g weight, measured during the still-holding phase). Predicted forces are 2.80 N (GT: 2.56 N) and 2.40 N (GT: 2.21 N), indicating some generalization to unseen contact geometries under this restricted setting.

C. Comparison with Baselines

To place NeuralActuator against principled model-based alternatives, we compare it with three classical sensorless force estimators on the weight benchmark: (i) *ID-Linear* [36], the textbook inverse-dynamics residual $\hat{\boldsymbol{\tau}}_{\text{ext}} = K_t \boldsymbol{i} + b - \boldsymbol{\tau}_{\text{ID}}(q, \dot{q}, \ddot{q})$ with per-joint K_t calibrated on free motion; (ii) *ID-Friction* [31], which augments (i) with Stribeck friction and backlash compensation; and (iii) the Generalized Momentum Observer (*GMO*) [11], which avoids $\ddot{q}$ via momentum integration with auto-tuned gain. All three baselines convert motor current to torque through the linear current–torque map and map the residual joint torque to end-effector force through the Jacobian pseudoinverse; importantly, they consume *ground-truth* robot states at every step, whereas NeuralActuator predicts forces from its own simulated rollout. Two observations stand out. First, fitting K_t per joint on the OpenManipulator-X already exposes the limits of the linear assumption: two joints yield negative slopes before clamping, consistent with the high gear ratio gearbox masking the underlying current–torque relation. Second, even after Stribeck and backlash compensation, the best classical method (GMO) trails NeuralActuator by $5.5\times$ in mean force MAE (Tab. VI, 0.66 N vs. 0.12 N). In the no-contact regime, classical methods produce 33–59% false-positive contact rates whereas NeuralActuator remains near zero. Adding Stribeck friction over the linear baseline provides only modest improvement (1.23 vs. 1.41 N), confirming parametric models cannot capture the nonlinearities of the geared servos used on this low-cost platform.

TABLE VI: **Force MAE (N) against model-based sensorless baselines** on the weight benchmark (lower is better). Baselines consume ground-truth states at every step; NeuralActuator uses simulated rollout states.

Method	Go Up & Stay			Pick & Place				Avg
	200g	300g	400g	200g	300g	400g	500g	
ID-Linear [36]	1.37	1.81	2.30	0.72	0.95	1.22	1.47	1.41
ID-Friction [31]	1.06	1.59	2.15	0.62	0.82	1.10	1.31	1.23
GMO [11]	0.58	0.66	1.23	0.33	0.47	0.63	0.75	0.66
NeuralActuator	**0.12**	**0.20**	**0.07**	**0.24**	**0.00**	**0.19**	**0.05**	**0.12**

D. Motor Condition Estimation

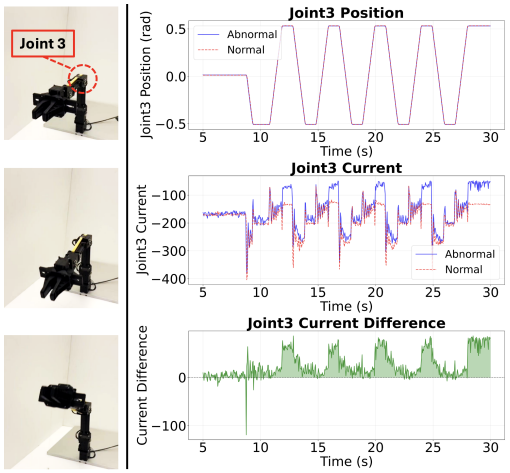
Next, we simulate motor degradation by wrapping rubber bands around joints to introduce additional mechanical resistance (Fig. 6 (a)). When commanding identical positions, the perturbed motor draws higher current to overcome the resistance while maintaining similar trajectories (Fig. 6 (b)). This consistent current deviation enables sensorless fault detection without additional sensors. In Tab. VII, we compare NeuralActuator with baselines for motor condition detection on a pick-and-place task with a 200 g object. NeuralActuator achieves the best performance with 91.0% accuracy with 96.2% recall and 0.95 AUC-ROC, demonstrating potential for predictive maintenance and fault detection on the platforms. Baseline details (Threshold (Thres.), SVM, Random Forest (RF)) are in Appendix L. The rubber band acts as a controlled proxy for mechanical degradation: it imposes a position-dependent resistance that raises the current required for the same trajectory, qualitatively matching the signature of increased bearing friction or gearbox wear. It does not capture intermittent electrical faults or coil-side failures, which we leave to future work.

TABLE VII: **Motor condition detection.**

Metric	Thres.	SVM	RF	Ours
Accuracy	58.6%	59.9%	67.1%	**91.0%**
Precision	0.0%	52.6%	62.3%	**84.5%**
Recall	0.0%	31.7%	52.4%	**96.2%**
AUC-ROC	0.45	0.62	0.72	**0.95**

E. Running time performance

Tab. VIII shows that NeuralActuator is efficient enough for simulation and real-time control: JAX JIT compilation yields sub-millisecond GPU inference latency and throughput well above the 60 Hz control rate. Batch processing further improves throughput, while the total parameter footprint remains small, enabling deployment on embedded platforms.



(a) (b)

Fig. 6: **Motor Condition Estimation.** (a) Rubber bands wrapped around Joint 3 to simulate increased resistance from motor degradation or wear. (b) Despite the same position trajectories (top), the abnormal motor requires a higher current (middle), yielding a detectable difference (bottom).

TABLE VIII: **Model parameters and runtime performance.**

Model Sizes			*Inference*		
Metric	Value	Unit	Metric	Value	Unit
Parameters	1.42M	–	Mean time	0.25	ms
FLOPs (fwd)	5.46M	–	P95 time	0.31	ms
Parameter memory	5.43	MB	Thru. (b=1)	4,019	Hz
			Thru. (b=32)	10,992	Hz

F. Sim-to-Real for Real Robot Control — Imitation Learning

To demonstrate the utility of NeuralActuator as a simulation component, we train a behavior cloning (BC) controller on top of a *frozen* neural actuator layer. Following [29], we collect expert demonstrations via teleoperation and train a policy network to predict commanded joint positions (see details in Appendix H). We train the policy using demonstrations with 100 g, 200 g, 300 g, and 500 g payloads, and evaluate on 400 g for object lift-and-hold and 500 g for pick-and-place. **Force-aware policy input.** In our setting, the BC policy is conditioned not only on proprioception and target joint positions, but also on the **external force signal** predicted by the frozen NeuralActuator module, $\hat{\mathbf{f}}_{\text{ext}}$. This augments position-only control with force feedback, which matters under varying payloads and intermittent contacts. The NeuralActuator module is frozen throughout policy training and the NAD data itself is collected through teleoperation rather than a learned controller, so the actuator mapping is identified from the hardware's input–output behavior and is not coupled to any specific control policy. **Position-only baseline.** The baseline trains the same policy without NeuralActuator and without any external force input. Tab. IX compares success rates with and without the neural actuator layer, showing that **accurate actuator modeling together with force feedback** consistently improves sim-to-real transfer for manipulation tasks (Fig. 1).

G. Visual Supervision

We demonstrate the application of our differentiable simulation framework with differentiable rendering for fully differentiable hand-eye calibration. We initialize T_{cb} (camera-from-base) with a geometric 3-point initialization that aligns depth-lifted image keypoints with the corresponding forward-kinematics keypoints, and jointly refine the camera extrinsics and per-frame joint configurations through differentiable silhouette-based optimization, following the EasyHeC [7] formulation; camera intrinsics are estimated using VGGT [45] and held fixed. The rendered robot silhouette is matched to a segmentation mask produced by SAM3 [6] with "robot arm" as the text prompt, and consistency is evaluated by mask IoU, reaching an average IoU of 0.8515. Please check out more details in Appendix D.

TABLE IX: **Behavior cloning success rates with NeuralActuator.** Results are averaged over 40 trials.

Task	**w/o NeuralActuator**	**w/ NeuralActuator**
Pick-and-Place	80%	**92.5%**
Go Up-and-Stay	85%	**95%**

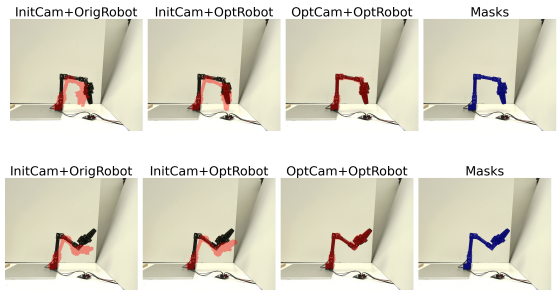


Fig. 7: **NeuralActuator for hand-eye calibration**. Combining NeuralActuator with a differentiable rendering pipeline for end-to-end hand-eye calibration.

H. Ablation on Network Architecture

Because there is no baseline specifically designed for this task, we evaluate the impact of different neural architectures for the actuator model, comparing three architectures with matched parameter counts (∼1.4M parameters) against our proposed approach. Let $\boldsymbol{x}_t \in \mathbb{R}^F$ denote the feature vector containing joint states, motor currents, and goals. We consider: (i) *MLP*: A feedforward network with LayerNorm predicting torques from flattened history $\boldsymbol{\tau}_t = f_\theta(\boldsymbol{x}_{t-L+1:t})$; (ii) *GRU* [9]: A recurrent model $\boldsymbol{h}_t = \text{GRU}_\theta(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$ with learnable initial states; (iii) *LSTM* [20]: Long short-term memory cells $(\boldsymbol{h}_t, \boldsymbol{c}_t) = \text{LSTM}_\theta(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}, \boldsymbol{c}_{t-1})$ for long-range dependencies. As shown in Tab. X, our Transformer-based approach achieves high dynamics modeling as well as force prediction accuracy compared with other design choices.

I. More Platforms

7-DoF Franka Emika Panda with joint torque sensors. This platform exposes per-joint torque measurements, enabling a *direct* predicted-vs-measured torque validation rather than the pose-only validation used on low-cost servos. For force estimation on Franka, the predicted joint torque $\hat{\tau}$ is fed back as an input feature to the force head. Across 200–600 g lift-and-hold trajectories, the model attains $\leq 3.7°$ per-joint MAE at all settings, 0.26–0.42 N force MAE, and **0.50–0.98 Nm torque MAE** against the on-board sensors (ground-truth torque range

TABLE X: **Comparison of neural architectures for actuation modeling**. We report rollout and force prediction accuracy at a 500-step prediction horizon. Bold marks the best result per column; ties are bolded jointly.

Model	J1	J2	J3	J4	Grip	F
MLP	4.55	5.81	3.48	2.16	0.91	0.47
GRU	1.83	**2.08**	**1.68**	1.70	**0.65**	0.49
LSTM	2.91	7.66	3.21	3.08	0.71	0.41
Ours	**1.78**	3.31	2.01	**1.58**	**0.65**	**0.23**

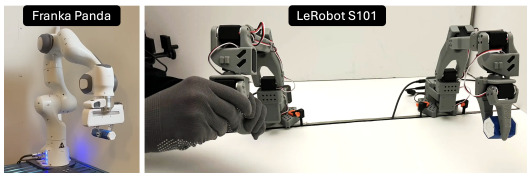


Fig. 8: **Cross-platform validation.** We train NeuralActuator from scratch on a 7-DoF Franka Emika Panda robot with on-board joint torque sensors and a 6-DoF SO-101 low-cost arm (STS3215 servos), spanning three actuator families and the \$500–\$30k+ cost spectrum.

0.1–30 Nm), giving the first sensor-grounded confirmation that NeuralActuator's torque predictions are physically meaningful. Per-joint breakdown at both 100- and 500-step rollouts is reported in Appendix J, Tab. A18.

6-DoF low-cost arm (SO-101). SO-101 uses STS3215 serial-bus servos with heterogeneous gearing (1:147–1:345), validating transfer beyond the Dynamixel family used in NAD. We collected 40,426 frames (649 s at 62.3 Hz) across six task–weight combinations following the NAD protocol. At a 500-step rollout, the model reaches 0.47–0.73 N force MAE for 300–500 g payloads. Joint MAE is larger than on the original platform (up to $\sim$9.8° on Joint 3; per-joint breakdown in Appendix J, Tab. A17), attributable to the heterogeneous gearing and the smaller per-platform dataset; the few-trajectory online adaptation routine of Appendix I mitigates both.

J. Learned Torque Values

We visualize the learned joint torques during real robot arm control. Fig. 9 compares the model-predicted torques with the measured motor currents over time for four joints. No torque sensor is available on this platform, so the comparison here is a qualitative illustration of the nonlinear current–torque relation learned by the model rather than a sensor-based validation; we report a direct predicted-vs-measured comparison on a 7-DoF arm with joint torque sensors in Sec. IV-I. Although the signals are roughly correlated, the effective current–torque gain and offset vary across joints and motion phases, exhibiting saturation, hysteresis, and noise. These patterns are consistent with low-cost actuator effects such as static friction, thermal drift, and gearbox backlash, under which simplified traditional models are often unstable or unidentifiable. In contrast, our *neural torque prediction* design allows the network to learn the full nonlinear and history-dependent mapping from telemetry to torque for more accurate actuation modeling.

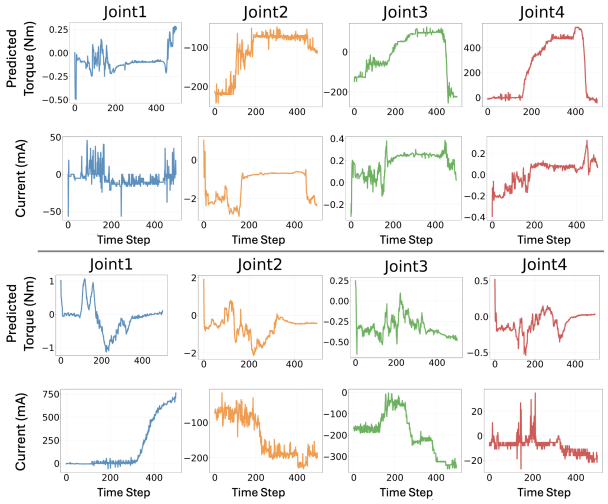


Fig. 9: **Relationship between motor current and predicted joint torque.** The effective gain and offset vary across joints and phases, with evident saturation and noise, indicating a distinctly nonlinear current-to-torque mapping. The averaged joint-angle MAE and gripper-position MAE are 1.26° and 1 mm, respectively (at 500 rollout steps). The y-axis values are model-*predicted* torques (Nm); the OpenManipulator-X has no on-board torque sensor, so this visualization illustrates the learned nonlinear current–torque mapping rather than a torque-sensor validation. A direct predicted-vs-measured comparison on a 7-DoF arm with joint torque sensors is in Sec. IV-I.

V. Conclusion

We presented NeuralActuator, a differentiable neural actuator model for low-cost robots that supports direct torque prediction and force perception without dedicated sensors. Instead of assuming a fixed linear current–torque relation, NeuralActuator learns a history-dependent mapping from telemetry and commands to torque, while jointly estimating external forces and contact. We provide a twin-arm teleoperation pipeline for collecting force-labeled data, a multi-task Transformer that predicts torques, forces, and contact states, and a torque-sensor-free training procedure through differentiable simulation. Experiments across three platforms spanning three actuator families and the \$500–\$30k+ cost spectrum show improved motion prediction, reliable force estimation on weight benchmarks, motor condition detection, and behavior cloning with NeuralActuator. *Limitations and Future Work.* Like all data-driven dynamics models, NeuralActuator accumulates error over long-horizon open-loop rollouts, eventually degrading tracking accuracy; the few-trajectory online adaptation routine in Appendix I mitigates this drift. Future work could explore multi-point force prediction and more scalable ways to learn force estimation from visual cues.

VI. Acknowledgements

This work has been done under the provisions of MIT-Amazon Science Hub. The authors would like to extend their sincere appreciation to Yuxiang Ma and Yuxin Song for their generous assistance with some of the hardware experiments. We are also deeply grateful to the anonymous reviewers for their thoughtful and constructive comments, which have improved the quality of the work described in this paper.

References

[1] Anurag Ajay, Jiajun Wu, Nima Fazeli, Maria Bauza, Leslie P Kaelbling, Joshua B Tenenbaum, and Alberto Rodriguez. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3066–3073. IEEE, 2018.

[2] Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond. *URL https://github. com/Genesis-Embodied-AI/Genesis*, 2024.

[3] Seyed Ali Baradaran Birjandi, Johannes Kühn, and Sami Haddadin. Observer-extended direct method for collision monitoring in robot manipulators using proprioception and imu sensing. *IEEE Robotics and Automation Letters*, 5(2):954–961, 2020.

[4] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

[5] Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Jess Moss, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. https://github.com/huggingface/lerobot, 2024.

[6] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts, 2025. URL https://arxiv.org/abs/2511.16719.

[7] Linghao Chen, Yuzhe Qin, Xiaowei Zhou, and Hao Su. Easyhec: Accurate and automatic hand-eye calibration via differentiable rendering and space exploration. *IEEE Robotics and Automation Letters*, 8(11): 7234–7241, November 2023. ISSN 2377-3774. doi: 10.1109/lra.2023.3315551. URL http://dx.doi.org/10.1109/LRA.2023.3315551.

[8] Peter Yichen Chen, Chao Liu, Pingchuan Ma, John Eastman, Daniela Rus, Dylan Randle, Yuri Ivanov, and Wojciech Matusik. Learning object properties using robot proprioception via differentiable robot-object interaction. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5997–6004. IEEE, 2025.

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[10] Jeremy A Collins, Cody Houff, Patrick Grady, and Charles C Kemp. Visual contact pressure estimation for grippers in the wild. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10947–10954. IEEE, 2023.

[11] Alessandro De Luca and Raffaella Mattone. Sensorless robot collision detection and hybrid force/motion control. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 999–1004. IEEE, 2005.

[12] Alessandro De Luca, Alin Albu-Schaffer, Sami Haddadin, and Gerd Hirzinger. Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *2006 IEEE/RSJ international conference on intelligent robots and systems*, pages 1623–1630. IEEE, 2006.

[13] Nolan Fey, Gabriel B Margolis, Martin Peticco, and Pulkit Agrawal. Bridging the sim-to-real gap for athletic loco-manipulation. *arXiv preprint arXiv:2502.10894*, 2025.

[14] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

[15] Wanjia Fu, Hongyu Li, Ivy X He, Stefanie Tellex, and Srinath Sridhar. Unitac: Whole-robot touch sensing without tactile sensors. *arXiv preprint arXiv:2507.07980*, 2025.

[16] Sami Haddadin, Alin Albu-Schaffer, Alessandro De Luca, and Gerd Hirzinger. Collision detection and reaction: A contribution to safe physical human-robot interaction. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3356–3363. IEEE, 2008.

[17] Linyan Han, Jianliang Mao, Pengfei Cao, Yahui Gan, and Shihua Li. Toward sensorless interaction force estimation for industrial robots using high-order finite-time observers. *IEEE Transactions on Industrial Electronics*, 69(7):7275–7284, 2021.

[18] Ryo Hanai, Yukiyasu Domae, Ixchel G Ramirez-Alpizar, Bruno Leme, and Tetsuya Ogata. Force map: Learning to predict contact force distribution from vision. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3129–3136. IEEE, 2023.

[19] Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9474–9481. IEEE, 2021.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[21] Taylor A Howell, Simon Le Cleac'h, Jan Brüdigam, Qianzhong Chen, Jiankai Sun, J Zico Kolter, Mac Schwager, and Zachary Manchester. Dojo: A differentiable physics engine for robotics. *arXiv preprint arXiv:2203.00806*, 2022.

[22] Jin Hu and Rong Xiong. Contact force estimation for robot manipulator using semiparametric model and disturbance kalman filter. *IEEE Transactions on Industrial Electronics*, 65(4):3365–3375, 2017.

[23] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *ICLR*, 2020.

[24] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[25] Maged Iskandar, Alin Albu-Schäffer, and Alexander Dietrich. Intrinsic sense of touch for intuitive physical human-robot interaction. *Science Robotics*, 9(93): eadn4008, 2024.

[26] Yifeng Jiang, Jiazheng Sun, and C Karen Liu. Data-augmented contact model for rigid body simulation. In *Learning for dynamics and control conference*, pages 378–390. PMLR, 2022.

[27] Sung-Woo Kim, Buyoun Cho, Seunghoon Shin, Jun-Ho Oh, Jemin Hwangbo, and Hae-Won Park. Force control of a hydraulic actuator with a neural network inverse model. *IEEE Robotics and Automation Letters*, 6(2): 2814–2821, 2021.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Masato Kobayashi, Thanpimon Buamanee, Yuki Uranishi, and Haruo Takemura. Ilbit: Imitation learning for robot using position and torque information based on bilateral control with transformer. *IEEJ Journal of Industry Applications*, 14(2):161–168, 2025.

[30] Hyeokjun Kwon, Sung-Woo Kim, and Hyun-Min Joe. Learning-based force control of twisted string actuators using a neural network-based inverse model. *IEEE Robotics and Automation Letters*, 2024.

[31] Magnus Linderoth, Andreas Stolt, Anders Robertsson, and Rolf Johansson. Robotic force estimation using motor torques and modeling of low velocity friction disturbances. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3550–3556. IEEE, 2013.

[32] Hongda Liu, Wentie Niu, and Yonghao Guo. Direct torque control for pmsm based on the rbfnn surrogate model of electromagnetic torque and stator flux linkage. *Control Engineering Practice*, 148:105943, 2024. doi: 10.1016/j.conengprac.2024.105943.

[33] Sichao Liu, Lihui Wang, and Xi Vincent Wang. Sensorless force estimation for industrial robots using disturbance observer and neural learning of friction approximation. *Robotics and Computer-Integrated Manufacturing*, 71:102168, 2021.

[34] Tribotix Pty Ltd. Pincherx 150 5dof robot arm. https://tribotix.com/product/pincherx-150-robot-arm/, 2020. Accessed: 2025-08-12.

[35] Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics. https://github.com/nvidia/warp, March 2022. NVIDIA GPU Technology Conference (GTC).

[36] Emanuele Magrini, Fabrizio Flacco, and Alessandro De Luca. Estimation of contact forces using a virtual force sensor. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2126–2133. IEEE, 2014.

[37] Lucas Manuelli and Russ Tedrake. Localizing external contact using proprioceptive sensors: The contact particle filter. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5062–5069. IEEE, 2016.

[38] Jonas Osburg, Ivo Kuhlemann, Jannis Hagenah, and Floris Ernst. Using deep neural networks to improve contact wrench estimation of serial robotic manipulators in static tasks. *Frontiers in Robotics and AI*, 9:892916, 2022.

[39] Interbotix (Trossen Robotics). Phantomx reactor robot arm (ax-12a). https://www.interbotix.com/p/phantomx-ax-12-reactor-robot-arm.aspx, 2025. Accessed: 2025-08-12.

[40] ROBOTIS Co., Ltd. Openmanipulator-x: Overview (rm-x52-tnm). https://emanual.robotis.com/docs/en/platform/openmanipulator_x/overview/, 2017. Accessed: 2025-07-12.

[41] Laura Schwendeman, Andrew SaLoutos, Elijah Stanger-Jones, and Sangbae Kim. Improving domain transfer of robot dynamics models with geometric system identification and learned friction compensation. In *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, pages 1–8. IEEE, 2023.

[42] Agon Serifi, Espen Knoop, Christian Schumacher, Naveen Kumar, Markus Gross, and Moritz Bächer. Transformer-based neural augmentation of robot simulation representations. *IEEE Robotics and Automation Letters*, 8(6):3748–3755, 2023.

[43] Mikko Tahkola, Janne Keränen, Denis Sedov, Mehrnaz Farzam Far, and Juha Kortelainen. Surrogate modeling of electrical machine torque using artificial neural networks. *IEEE Access*, 8:220027–220045, 2020. doi: 10.1109/ACCESS.2020.3042834.

[44] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

[45] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings*

of the Computer Vision and Pattern Recognition Conference, pages 5294–5306, 2025.

[46] Songyi Wang and Xinjian Wang. A pinn-based nonlinear pmsm electromagnetic model using differential inductance theory. *Applied Sciences*, 15(13):7162, 2025. doi: 10.3390/app15137162.

[47] Yufei Wang, David Held, and Zackory Erickson. Visual haptic reasoning: Estimating contact forces by observing deformable object interactions. *IEEE Robotics and Automation Letters*, 7(4):11426–11433, 2022.

[48] Jie Xu, Eric Heiden, Iretiayo Akinola, Dieter Fox, Miles Macklin, and Yashraj Narang. Neural robot dynamics. *arXiv preprint arXiv:2508.15755*, 2025.

[49] Yu-Bai Yan, Jia-Ning Liang, Tian-Fu Sun, Jian-Ping Geng, Gang Xie, and Dong-Jia Pan. Torque estimation and control of pmsm based on deep learning. In *2019 22nd International Conference on Electrical Machines and Systems (ICEMS)*. IEEE, 2019. doi: 10.1109/ICEMS.2019.8921886.

[50] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319, 2020.

[51] Peiyuan Zhi, Peiyang Li, Jianqin Yin, Baoxiong Jia, and Siyuan Huang. Learning a unified policy for position and force control in legged loco-manipulation. In *Conference on Robot Learning*, pages 652–669. PMLR, 2025.