

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

Zhiyang Dou¹, John U. Onyemelukwe^{1*}, Hangxing Zhang^{1*}, Heng Zhang¹, Minghao Guo¹, Yunsheng Tian¹, Michal Piotr Lipiec¹, Joshua Jacob¹, Chao Liu¹, Peter Yichen Chen¹, Yuri Ivanov^{2,†} and Wojciech Matusik¹

¹MIT ²Amazon Robotics

Abstract—Differentiable simulators have advanced policy learning and model-based control across diverse robotic tasks. To date, actuator dynamics remain underexplored and can be a major source of sim-to-real error, especially on low-cost platforms where the linear current-to-joint-torque approximation $\tau = K_t I$ becomes unreliable under commanded-target tracking because of friction, hysteresis, backlash, and thermal effects. Beyond forward dynamics, accurate actuator models also support force perception and integrated force/position control in manipulation tasks. We present NeuralActuator, a neural actuator model that jointly predicts (i) a simulator-equivalent generalized-effort surrogate for trajectory propagation on low-cost servo platforms, (ii) external forces together with a contact-probability gate for sensorless force perception, and (iii) a motor-condition score for the supervised joint, distinguishing normal from mechanically restricted operation. We introduce a twin-arm teleoperation system that collects robot states and actuator telemetry alongside external-force labels, yielding the Neural Actuation Dataset (NAD). The torque-surrogate head is trained through differentiable simulation from pose trajectories without direct generalized-effort labels; the force, gate, and motor-condition heads use direct supervision from their respective labels. 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—spanning three actuator families and costs from approximately \$500 to more than \$30,000. The low-cost platforms support dynamics and force evaluation, while the offline Franka experiment provides an additional payload-force-estimation benchmark; the experiments also demonstrate its application for motor condition estimation and improved behavior-cloning performance when NeuralActuator is used as a pretrained module. Project Page: <https://frank-zy-dou.github.io/projects/NeuralActuator/index.html>.

I. INTRODUCTION

Differentiable simulators [24, 36, 13, 22, 15] 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, actuator dynamics remain comparatively underexplored. A common control approximation assumes that joint torque scales linearly with current, e.g., $\tau = K_t I$. Although reasonable for well-calibrated industrial actuators, this approximation becomes unreliable on cost-effective, servo-driven platforms [4, 40, 35, 41]. These platforms often use mechanically commutated actuators (e.g., coreless brushed DC servo modules) whose hardware simplicity is accompanied by significant non-idealities: gearbox

*These authors contributed equally. Both were research assistants at MIT CDFG.

†This work is unrelated to the author’s position at Amazon.

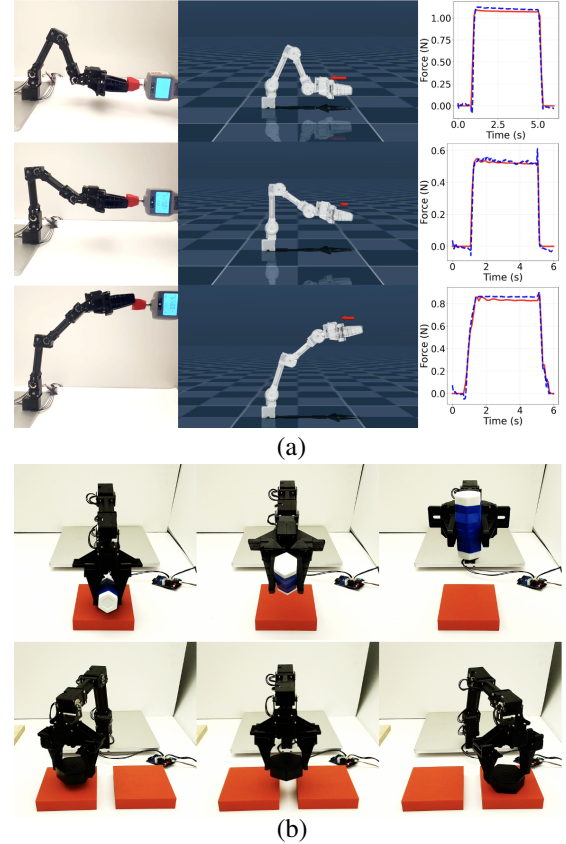


Fig. 1: (a) **Push-force gauge validation at three end-effector heights.** We show NeuralActuator pushing a force gauge at *low*, *middle*, and *high* end-effector heights (top to bottom). Left: real-robot execution; middle: corresponding simulated rollout; right: contact-axis force-magnitude profiles. The trace-level mean absolute errors (MAEs) are 0.037 N, 0.015 N, and 0.028 N, respectively. These values correspond to the three representative front-push sequences shown; Tab. VI reports aggregate componentwise force errors across positions, front/top contact directions, and rollout horizons. The $---$ shows the predicted magnitude, and the $---$ shows the force-gauge ground truth. (b) **Estimated forces for downstream tasks.** NeuralActuator provides online force feedback to behavior-cloning controllers for real-robot object lifting and holding (top) and pick-and-place (bottom) tasks.

friction, hysteresis, backlash, current saturation, and thermal drift affect joint-side effort. Because the relevant internal friction and transmission states are inaccessible, these time-varying effects resist analytical modeling, resulting in poor

effort tracking, unstable impedance behavior, and degraded sim-to-real transfer.

Beyond accurate modeling, a reliable actuator model also supports *force perception without dedicated force/torque sensors at inference time*. Recent studies demonstrate torque-based sensing for e-skin-like contact perception [26], proprioceptive estimation in legged robots [14], and force estimation and control in manipulation [29, 19, 9, 52]. Such torque-based methods, however, generally rely on high-quality effort feedback; when it is unavailable on low-cost platforms, an unreliable current–torque mapping undermines analogous force inference. Because physical interaction is mediated by force, contact-force prediction can support manipulation without precise geometric models 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 can be integrated with a differentiable simulator, enabling end-to-end training via gradient-based supervision from real-robot rollouts. NeuralActuator learns a history-dependent mapping from command and state histories together with actuator-side telemetry, including the platform-specific effort signal, voltage, and temperature where available, to simulator-equivalent generalized-effort surrogates (termed torque surrogates below), external forces, contact probabilities, and a motor-condition score that distinguishes normal from mechanically restricted operation on the supervised joint. Prior actuator models use torque targets obtained either from joint-torque sensing [25] or from calibrated motor-current estimates [42]; both forms of supervision are difficult to obtain reliably on many low-cost servos. Instead, we supervise the torque-surrogate head using pose trajectories: forward integration through our differentiable dynamics maps the predicted generalized inputs to future configurations, making pose an effective surrogate signal. The force, contact-gate, and motor-condition heads use direct supervision. To capture nonlinear, time-varying actuator effects, NeuralActuator adopts a Transformer architecture that processes recent state-and-command sequences through multi-head self-attention, modeling temporal dependencies and inter-joint coupling. It outputs per-actuator generalized-effort surrogates, external forces, contact probabilities, and motor-condition scores 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-force labels. 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 synchronized supervision 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 [4], and a 7-DoF Franka Emika Panda industrial arm (Sec. IV-F)—spanning three actuator families and costs from approximately \$500 to more than \$30,000. We evaluate modeling fidelity on the low-cost arms, cross-platform external-force estimation including an offline Franka benchmark, online adaptation, throughput, and downstream real-robot control. On held-out rollouts, NeuralActuator yields low rollout errors (Sec. IV-A), accurate external-force estimates (Sec. IV-B), and high Joint 3 condition-classification accuracy (Sec. IV-G). The evaluated implementation provides sub-millisecond GPU inference at the 60 Hz control rate with a modest parameter footprint (Sec. IV-H). We also evaluate online adaptation using 12 newly collected trajectories (Sec. IV-E). For downstream real-robot control, NeuralActuator provides online force feedback to behavior-cloning policies for object manipulation. We show that learning actuator behavior together with external force perception improves task success rates (Sec. IV-I). Finally, we demonstrate that combining differentiable simulation with silhouette-based image-space supervision provides a visual training signal for NeuralActuator (Sec. IV-J). 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-surrogate prediction, with additional outputs for external force prediction, contact detection, and motor-condition estimation—enabling force perception without dedicated force/torque sensors at inference time—and a training scheme that leverages differentiable physics without direct generalized-effort labels.
- 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 external-force labels, enabling training and benchmarking for low-cost actuator modeling.
- Experiments across three platforms evaluate actuator-dynamics modeling, sensorless force estimation, Joint 3 condition estimation, downstream manipulation, and computational efficiency.

II. RELATED WORK

A. Learning Neural-Augmented Simulation

To close the sim-to-real gap, prior work has augmented analytical simulators with learned residual models; for example, the approach in [1] uses a stochastic RNN to capture

¹https://emmanual.robotis.com/docs/en/platform/openmanipulator_x/overview/

residual dynamics and align simulation with reality. Jiang et al. [27] augment a rigid-body simulator with a data-driven contact model that predicts aggregate 3D contact impulses while retaining analytical contact constraints. TossingBot [51] further combines an analytic ballistic prior with a neural residual learned from visual features. NeuralSim [20] 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. [43] use a Transformer to predict residual state corrections for actuators and rigid bodies from simulated states and interaction forces. Recently, Xu et al. [49] present NeRD, which learns robot-specific dynamics models for predicting future states for articulated rigid bodies under contact constraints. Hwangbo et al. [25] train a supervised actuator network from physical-system data and use it in the simulation loop to model each joint of ANYmal; the network maps state histories and position targets to joint torques for simulation. Schwendeman et al. [42] introduce 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 are estimated from motor-current measurements. Accordingly, these approaches use torque targets obtained either from joint-torque sensing [25] or from a calibrated current–torque estimate [42]; both forms of supervision are difficult to obtain reliably on many low-cost servo platforms. In contrast, the torque-surrogate head in NeuralActuator requires no direct generalized-effort labels: it is supervised through differentiable simulation using pose trajectories, without assuming reliable current–torque calibration. Concurrently, Fey et al. [12] learn residual corrective torques via reinforcement learning from tracking errors; NeuralActuator instead predicts simulator-equivalent generalized-effort surrogates without assuming a fixed current–torque conversion, while its separate force head receives direct supervision from contact-force labels.

a) Learning Actuator Models: Existing work mostly targets permanent-magnet electrical machines, using neural models to estimate, control, or emulate electromagnetic torque from electrical states or simulation-derived features [50, 32, 47, 44]. These approaches concern electronically commutated machine models rather than the low-cost integrated coreless DC servo dynamics studied here. Prior work also covers hydraulic actuators [28] and twisted-string actuators [30]. Unlike these efforts, NeuralActuator targets low-cost *coreless DC* servo actuators, where friction, backlash, saturation, and thermal drift can make the effective current–torque relationship nonlinear and history-dependent.

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 [10, 11, 17]. Building on reliable effort feedback, recent systems further elevate this idea to richer “intrinsic” contact perception (e.g., e-skin–like touch with-

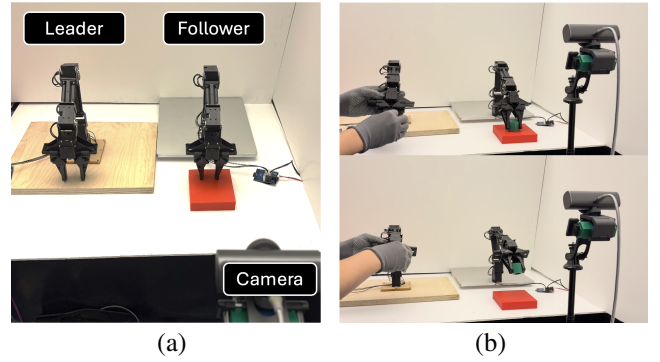


Fig. 2: **Neural Actuation Dataset (NAD) data collection.** (a) Overview of the twin-arm leader–follower system with an external camera. (b) An operator kinesthetically drives the leader to generate diverse motions; the follower mirrors these motions while manipulating a payload and recording synchronized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

out tactile skins) [26], whole-robot touch sensing without dedicated tactile sensors [14], and learning-based controllers that couple force and motion in manipulation and locomotion [29, 52]. 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–torque mappings.

b) External force estimation on robot arms: For manipulators, a complementary body of work 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 [37, 38], as well as observer and filtering formulations that improve robustness under friction and model uncertainty [23, 33, 18, 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 [39]. Chen et al. [7] 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 is provided in Tab. A20 in the Appendix.

III. METHOD

A. Problem Formulation

Consider a robotic manipulator with n_a arm joints and n_m modeled actuator channels, including a gripper channel when applicable. For OpenManipulator-X, $n_a = 4$ and $n_m = 5$. Let τ_j^{real} denote the physical joint-side effort for actuator channel j (torque for a revolute joint). It depends on the commanded position, effort-related actuator telemetry, state, and operating history:

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

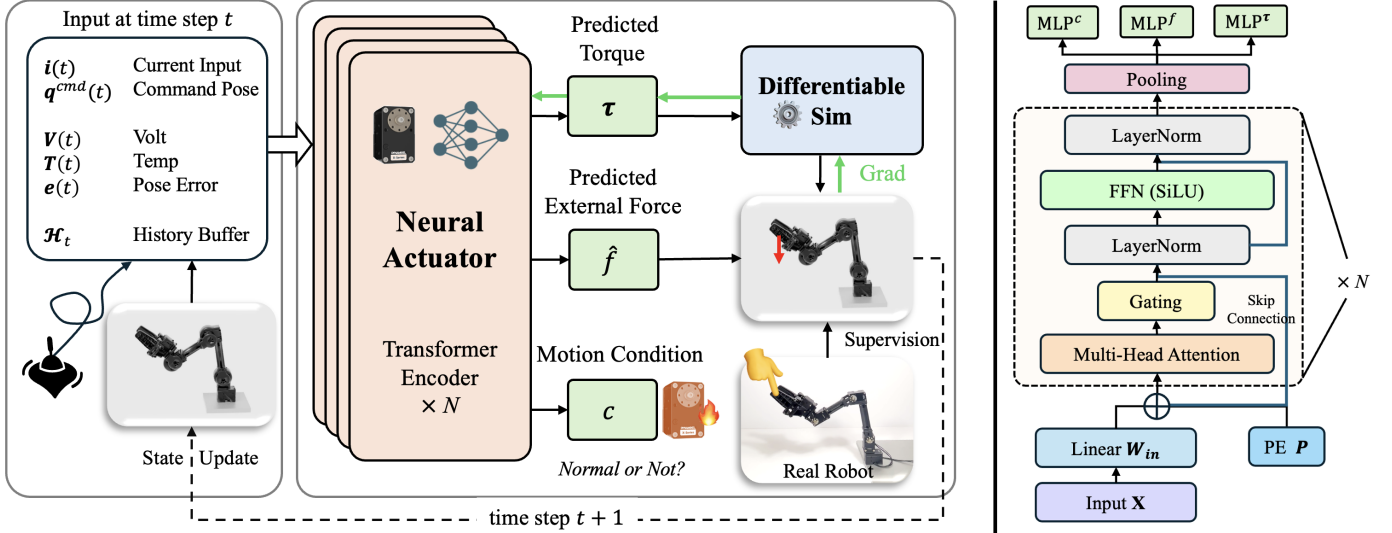


Fig. 3: **NeuralActuator pipeline.** At each time step t , NeuralActuator takes the commanded pose $q^{\text{cmd}}(t)$, an effort-related actuator signal $u(t)$, actuator-side telemetry, and tracking feedback $e(t)$, together with a history buffer $\mathcal{H}_t$ that summarizes recent commands, states, and telemetry (e.g., joint poses and velocities). A Transformer encoder ($\times N$ blocks) maps the sequence to four task heads: (i) a simulator-equivalent generalized-effort surrogate τ , termed the torque surrogate below, (ii) a raw external-force estimate $\hat{\mathbf{f}}_{\text{raw}}$, (iii) a contact-probability gate g , and (iv) a per-motor condition vector $\mathbf{c}$. The gated force is $\hat{\mathbf{f}} = g\hat{\mathbf{f}}_{\text{raw}}$. Only the clipped predicted surrogate $\bar{\tau}$ is supplied to the differentiable simulator as its generalized control input; the simulator then advances the state to time $t+1$ and updates the history buffer. The force, gate, and condition predictions are supervised outputs and do not enter the simulator state update. Gradients through *Diff Sim* supervise the torque-surrogate head from real-robot state transitions, while the other heads use their direct losses. **Right:** High-level schematic of the Transformer and output heads. The implementation uses the pre-normalized gated-attention and GELU encoder blocks specified in the text; the condition branch represents per-motor condition scores, and the separately implemented contact gate is omitted for visual clarity.

where $q_j^{\text{cmd}}(t)$ is the commanded (goal) position, $q_j(t)$ and $\dot{q}_j(t)$ are joint position and velocity, and $V_j(t)$ and $T_j(t)$ denote the supply voltage and thermal state, respectively. The platform-specific signal u_j is motor current on OpenManipulator-X, the signed load register on SO-101, and commanded joint torque on Franka. Let $\mathbf{x}_t$ collect the command, proprioceptive state, and actuator telemetry at time t . The model used in our experiments takes the eight preceding feature vectors $\mathcal{H}_t = \{\mathbf{x}_{t-8}, \dots, \mathbf{x}_{t-1}\}$ and appends the current vector $\mathbf{x}_t$, yielding the nine-token sequence $\mathbf{X}_t = (\mathbf{x}_{t-8}, \dots, \mathbf{x}_t)$.

In parallel with the learned torque-surrogate output, we predict external forces and motor-condition scores from the same input features:

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

$$\mathbf{c}(t) = f_{\text{cond}}(\mathbf{X}_t) \in [0, 1]^{n_m} \quad (3)$$

where $\hat{\mathbf{f}}^{\text{ext}}(t)$ is the predicted end-effector external force. The architecture emits one condition score $c_j(t)$ per actuator channel. In the OpenManipulator-X condition benchmark, only the Joint 3 component is supervised and evaluated: c_3 estimates the probability of normal operation, and $c_3 = 0$ denotes the mechanically restricted condition. We do not assign a calibrated probabilistic interpretation to the unsupervised components in this experiment.

B. Physical Interpretation through Manipulator Dynamics

The manipulator equation motivates separate actuation and interaction outputs, but our implementation does not impose

it as a decomposition constraint. We define $\mathbf{f}^{\text{ext}}$ as the force exerted by the environment on the robot, matching the sign of the recorded force labels, and write

$$\begin{aligned} \tau_{\text{ID}} &= M(q)\ddot{q} + C(q, \dot{q})\dot{q} + \mathbf{g}(q) \\ &= \tau_{\text{act}} + \tau_{\text{ext}}, \\ \tau_{\text{ext}} &= J_v(q)^\top \mathbf{f}^{\text{ext}}. \end{aligned} \quad (4)$$

Here, $q \in \mathbb{R}^{n_a}$ contains the arm coordinates. The quantities $M(q)$, $C(q, \dot{q})\dot{q}$, and $\mathbf{g}(q)$ are the analytical rigid-body terms encoded by the simulator, and τ_{act} denotes the physical joint-side actuator torque. The translational Jacobian $J_v(q)$ is evaluated at the force-reference point; using a three-dimensional force rather than a six-dimensional wrench assumes that the residual moment about that point is negligible. The gripper geometry is described separately in Sec. III-E. This convention gives $\tau_{\text{ext}} = \tau_{\text{ID}} - \tau_{\text{act}}$, which is the residual used by the classical force-estimation baselines. Those methods approximate the actuator term by a linear current–torque model, $\tau_{\text{act}} = \mathbf{K}_t \mathbf{i} + \mathbf{b}$, where $\mathbf{K}_t = \text{diag}(K_{t,1}, \dots, K_{t,n_a})$, and map the resulting residual to Cartesian force. On geared low-cost servos, however, friction, backlash, saturation, and thermal drift make this linear map unreliable; its errors therefore contaminate the inferred contact force. NeuralActuator instead predicts a simulator-equivalent generalized-effort surrogate τ^{pred} , termed the torque surrogate below, and an external-force estimate $\hat{\mathbf{f}}^{\text{ext}}$ with dedicated heads. During rollout, only the clipped surrogate $\bar{\tau} = \text{clip}(\tau^{\text{pred}})$ drives the simulator; the force head is directly supervised and is not mapped through $J_v(q)^\top$ into the state update. Consequently, on pose-supervised platforms, $\bar{\tau}$ is trained to represent the generalized

input required to reproduce the observed motion; τ^{pred} is its pre-clipping network output. For a revolute channel, the applied surrogate coincides with the physical joint-side actuator torque only when external loads are absent or represented in the forward model and the remaining dynamics are modeled accurately; otherwise, unmodeled contact, payload effects, and dynamics mismatch may be absorbed into it. In the reported OpenManipulator-X contact and payload experiments, both training and evaluation use the same unloaded robot model with MuJoCo/MJX contact dynamics disabled; no measured or predicted external forces, payload bodies, or link-mass changes are supplied to the forward model. The applied surrogate is therefore a lumped simulator-equivalent generalized input rather than an identifiable estimate of actuator torque in those experiments. Equation 4 therefore provides physical context rather than an enforced, identifiable decomposition of the actuation and force outputs. The condition head in Eq. 3 provides auxiliary architecture-level scores from the shared command, state, and telemetry history; in the reported benchmark, only the supervised Joint 3 score is used as a diagnostic.

C. Neural Actuation Dataset (NAD)

To learn the actuator model, we first collect synchronized robot-state, actuator-telemetry, and force data using a twin-arm teleoperation system built on two identical OpenManipulator-X robots [41]. 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).

The *commanded target joint positions* sent to the follower’s actuators are set to the leader’s 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, interaction force is measured with a fixture-mounted six-axis force/torque sensor. We use its three force channels as ground-truth end-effector contact-force measurements $\mathbf{f}^{\text{et}}(t) \in \mathbb{R}^3$; the three moment channels are not used for supervision. Before training, force labels are transformed into the robot base frame and retain the environment-on-robot sign convention used in Eq. 4.

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. Together, these streams support pose-based surrogate learning and direct force supervision. The OpenManipulator-X model-development and evaluation subset used in our experiments comprises the following three components:

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.

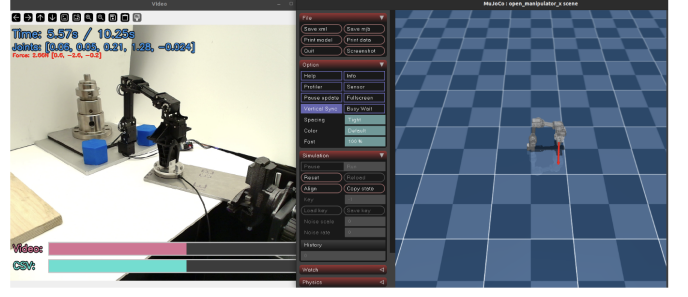


Fig. 4: **Data verification in NAD.** We time-synchronize the real-robot video (left) with the logged trajectory and verify alignment in a GUI (right), where the red arrow visualizes the estimated external force.

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 experimental subset uses payloads with $m \in \{200, 300, 400, 500\}$ g; the full NAD dataset additionally contains 100 g payload trajectories. During *loaded/grasped* intervals, we assign the nominal base-frame load $\mathbf{f}_{\text{grav}}^{\text{gt}} = m \mathbf{g}_{\text{base}} = [0, 0, -mg]^\top$. This reference is exact for static or quasi-static holding; during motion, it provides nominal supervision that does not account for object inertia or gripping friction. 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 frame to capture full 3D force components arising from both dominant-axis interactions and kinematic coupling (74,020 frames, avg. 10.50 s).

We also collect data for the cross-platform evaluation in Sec. IV-F, using an SO-101 arm logged via the LeRobot stack [4] and a 7-DoF Franka Emika Panda industrial arm. Franka provides an additional offline payload-force-estimation benchmark in the reported results.

Mechanically restricted motor operation.

In this task, we constrain Joint 3 with rubber bands to introduce additional mechanical resistance; see the inset. Under this asymmetric restriction, we collect *pick and place* trajectories with and without a 200 g payload (49,898 frames, avg. 21.20 s), enabling disentanglement of contact-induced and condition-induced deviations.

We split the dataset into training, validation, and test sets in an 8:1:1 ratio. All data in NAD are calibrated and human-

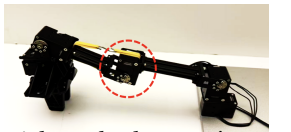


TABLE I: **OpenManipulator-X model-development and evaluation subset.** The reported 94.52 minutes sum the task-assignment durations used in the experiments; nominal trajectories reused for motor-condition comparison also appear in the free-motion and force-labeled categories. See Appendix A for a detailed task-level breakdown.

Component	Description	Duration (min)
Free motion	No external force	~ 34.15
Force-labeled	Known payloads or measured force	~ 46.24
Motor condition	Mechanically restricted	~ 14.13
Total		~ 94.52

verified as shown in Fig. 4. See Appendix A for additional details.

D. NeuralActuator

To capture temporal dependencies and nonlinear actuator dynamics, we employ a Transformer encoder that processes recent commands, states, and telemetry to produce multi-task actuator and interaction estimates. In NeuralActuator, the input sequence $\mathbf{X}_t \in \mathbb{R}^{L \times n_f}$ stacks per-time-step feature vectors $\mathbf{x}_t \in \mathbb{R}^{n_f}$. It comprises $H_{\text{hist}} = 8$ historical frames followed by the current frame, so $L = H_{\text{hist}} + 1 = 9$. In the primary low-cost configurations, each $\mathbf{x}_t$ aggregates three families of signals—(i) commanded targets ($q^{\text{cmd}}, q_g^{\text{cmd}}$), (ii) proprioception ($q, \dot{q}, q_g$), and (iii) actuator telemetry (u, V, T)—together with tracking-error features ($e = q^{\text{cmd}} - q$, $e_g = q_g^{\text{cmd}} - q_g$). All features are normalized before being fed to the network. The Franka experiment uses the platform-specific feature vector described in Appendix B. In simulation, the gripper feature q_g is the single-finger prismatic-joint coordinate; it is distinct from the total physical jaw opening $a(\alpha)$ in Sec. III-E.

An input projection and learned positional embeddings initialize $\mathbf{H}_{\text{enc}}^{(0)} = \mathbf{X}_t \mathbf{W}_{\text{in}} + \mathbf{P}$, where $\mathbf{W}_{\text{in}} \in \mathbb{R}^{n_f \times d}$ and $\mathbf{P} \in \mathbb{R}^{L \times d}$.

We use pre-normalized residual blocks with a query-dependent elementwise gate on the attention output. For layer l ,

$$\mathbf{U}^{(l)} = \text{LayerNorm}\left(\mathbf{H}_{\text{enc}}^{(l-1)}\right), \quad (5)$$

$$\mathbf{A}^{(l)} = \left[\text{MHA}\left(\mathbf{U}^{(l)}\right) \odot \sigma\left(\mathbf{U}^{(l)} \mathbf{W}_g^{(l)}\right) \right] \mathbf{W}_o^{(l)}, \quad (6)$$

$$\mathbf{Z}^{(l)} = \mathbf{H}_{\text{enc}}^{(l-1)} + \mathbf{A}^{(l)}, \quad (7)$$

$$\mathbf{H}_{\text{enc}}^{(l)} = \mathbf{Z}^{(l)} + \text{FFN}_{\text{GELU}}^{(l)}\left(\text{LayerNorm}\left(\mathbf{Z}^{(l)}\right)\right). \quad (8)$$

Here, $\mathbf{W}_g^{(l)}, \mathbf{W}_o^{(l)} \in \mathbb{R}^{d \times d}$, $\mathbf{U}^{(l)} \mathbf{W}_g^{(l)}$ is produced alongside the query projection, MHA denotes the concatenated per-head attention values before the output projection, and $\odot$ denotes elementwise multiplication. After the final encoder layer and layer normalization, temporal mean pooling yields $\mathbf{h}_{\text{pool}} \in \mathbb{R}^d$. The configuration uses four encoder layers, hidden dimension $d = 192$, four attention heads, and feed-forward dimension $d_{\text{ff}} = 384$. Four output heads with SiLU hidden activations predict a generalized-effort surrogate, raw force, contact probability, and per-motor condition scores.

1. Torque-Surrogate Prediction. NeuralActuator predicts a simulator-equivalent generalized-effort surrogate as

$$\boldsymbol{\tau}^{\text{pred}}(t) = g_{\theta}(\mathbf{X}_t), \quad (9)$$

where $\boldsymbol{\tau}^{\text{pred}}(t) \in \mathbb{R}^{n_m}$ denotes the pre-clipping network output and $\mathbf{X}_t = (\mathbf{x}_{t-8}, \dots, \mathbf{x}_t)$ contains proprioception, commands, and actuator telemetry. For the primary torque-actuated simulators, we use *torque surrogate* as shorthand for this generalized effort: it is torque-valued on revolute channels and force-valued on a prismatic gripper channel. The Franka configuration instead retains the robot model’s stock affine PD position actuators; on that platform, the corresponding head supplies a simulator-control surrogate rather than a directly torque-valued input. This design accommodates a nonlinear, history-dependent relationship between actuator telemetry and the effective generalized input required by the simulator. The surrogate therefore avoids relying on a fixed linear current–torque prior.

A single MLP head jointly predicts the surrogate values for all n_m actuator channels: $\boldsymbol{\tau}^{\text{pred}} = \text{MLP}_{\tau}(\mathbf{h}_{\text{pool}}) \in \mathbb{R}^{n_m}$, where MLP_{τ} uses SiLU hidden activations.

2. External Force Prediction. We predict end-effector external forces using a two-stage approach that decouples force regression from contact detection. A force-specific hidden feature is first computed from the shared Transformer representation. Separate linear readouts from this feature predict the raw 3D force $\hat{\mathbf{f}}_{\text{raw}} \in \mathbb{R}^3$ and the contact gate $g \in [0, 1]$. No dedicated force sensor is required at inference time. The gate represents the probability of physical contact, and the final force output is $\hat{\mathbf{f}} = g \hat{\mathbf{f}}_{\text{raw}}$. During training, the gate target is $g_{\text{gt}} = \mathbb{I}[\|\mathbf{f}_{\text{gt}}\|_2 > \epsilon]$. We set $\epsilon = 0.01$ N solely to distinguish zero from nonzero supervision labels.

3. Motor Condition Estimation. A separate head predicts the per-motor condition vector $\mathbf{c} = \sigma(\text{MLP}_{\text{cond}}(\mathbf{h}_{\text{pool}})) \in [0, 1]^{n_m}$. The OpenManipulator-X benchmark supervises and evaluates only the Joint 3 component from the shared command, state, and telemetry history: a value near one indicates normal operation, whereas a value near zero indicates the mechanically restricted condition. The remaining output components are not interpreted as calibrated condition probabilities in this experiment.

Differentiable Simulation Integration. NeuralActuator is integrated with differentiable physics simulators. We embed the model within the simulation loop to enable gradient-based optimization through the system dynamics. When differentiable rendering is enabled, image-space supervision follows the same computational path: the silhouette loss is backpropagated through the renderer, simulated configurations, and differentiable dynamics to the parameters of NeuralActuator. *Forward Dynamics.* For the torque-actuated OpenManipulator-X and SO-101 simulators, the torque-surrogate head evaluates at time step t $\boldsymbol{\tau}_t^{\text{pred}} = g_{\theta}(\mathbf{X}_t)$. During training, platform-specific generalized-effort bounds are applied before simulation, $\bar{\boldsymbol{\tau}}_t = \text{clip}(\boldsymbol{\tau}_t^{\text{pred}}; \boldsymbol{\tau}_{\min}, \boldsymbol{\tau}_{\max})$, and the bounded surrogate advances

$$\mathbf{s}_{t+1} = F_t(\mathbf{s}_t, \bar{\boldsymbol{\tau}}_t) = \text{DiffSim}(\mathbf{s}_t, \bar{\boldsymbol{\tau}}_t). \quad (10)$$

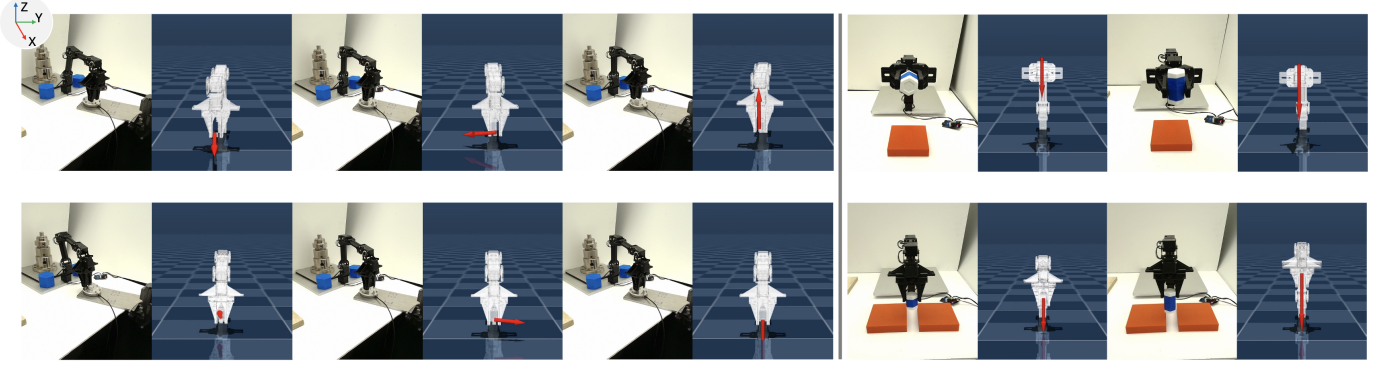


Fig. 5: **Robot arm simulation with external force estimation on the test set.** (a) NeuralActuator estimates end-effector external forces along six directions ($\pm X, \pm Y, \pm Z$); red arrows visualize the estimated force vectors in simulation (paired with corresponding real-robot executions). (b) 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). Additional examples are shown in the accompanying video.

Here, $\mathbf{s}_t = [q_t, \dot{q}_t]$ is the simulated joint state. The Franka configuration instead supplies the simulator-control surrogate to its affine PD position actuators, as described above. The force, gate, and condition outputs receive their paired direct labels at each training-rollout step but do not enter F_t or impose a dynamics-based decomposition on τ^{pred} . Offline training and evaluation use the complete time-aligned effort-telemetry sequence recorded with each trajectory; the resulting multi-step evaluations are therefore recorded-telemetry-conditioned rollouts rather than forecasts of the actuator response to counterfactual future commands. For online OpenManipulator-X deployment, prediction is strictly causal. At each control cycle, the robot reads the current state and available actuator telemetry, including the live measured current $\mathbf{i}_t$; NeuralActuator then estimates the current surrogate, force, and condition scores, and a feedback controller can use $\hat{\mathbf{f}}_t$ to generate the next command. After inference, $\mathbf{x}_t$ is appended to the history used at the next control cycle; $\mathbf{x}_{t+1}$ enters only after the next telemetry packet arrives. Online predictions therefore use no future current measurements. Planning over counterfactual future commands would additionally require a model of the corresponding effort signal.

Gradient Computation. For $\mathcal{L} = \sum_{k=1}^T \ell_k(\mathbf{s}_k)$, automatic differentiation accounts for the effect of τ_t^{pred} on $\mathbf{s}_{t+1}$ and performs backpropagation through time according to

$$\frac{d\mathcal{L}}{d\theta} = \sum_{k=1}^T \frac{\partial \ell_k}{\partial \mathbf{s}_k} \frac{d\mathbf{s}_k}{d\theta}, \quad (11)$$

$$\frac{d\mathbf{s}_{t+1}}{d\theta} = \frac{\partial F_t}{\partial \mathbf{s}_t} \frac{d\mathbf{s}_t}{d\theta} + \frac{\partial F_t}{\partial \tau_t} \frac{d \text{clip}(g_\theta(\mathbf{X}_t))}{d\theta}. \quad (12)$$

The total derivative includes the dependence of the simulated-state components in $\mathbf{X}_t$ on earlier predictions; consequently, losses at later steps backpropagate to preceding torque-surrogate predictions wherever the clipping and simulator sensitivities are nonzero. Equivalently, the Markov state for this recurrence is the augmented state $(\mathbf{s}_t, \mathcal{H}_t)$; Eq. 12 suppresses the explicit history-buffer component for readability.

Model Training. Since direct generalized-effort labels are unavailable on the low-cost platforms, the torque-surrogate

head is supervised through pose trajectories: forward integration converts the predicted generalized inputs into future configurations, making measured poses effective surrogate targets. The force, gate, and motor-condition heads receive direct supervision from the labels paired with each training-rollout input wherever those labels are available. We use Smooth L1 (Huber) regression losses. The following equations describe the primary OpenManipulator-X configuration. Let Huber_β denote the elementwise Huber penalty with transition parameter β , and let $v_{t,k} \in \{0, 1\}$ be the optional validity mask for force channel k . The sets $\mathcal{T}_f$ and $\mathcal{T}_c$ index the input-label pairs constructed by the training rollout for the directly supervised heads. The regression terms are

$$\mathcal{L}_{\text{joint}} = \frac{1}{|\mathcal{T}|n_a} \sum_{t \in \mathcal{T}} \sum_{j=1}^{n_a} \text{Huber}_1(q_{t,j}^{\text{sim}} - q_{t,j}^{\text{real}}), \quad (13)$$

$$\mathcal{L}_{\text{grip}} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \text{Huber}_1(\kappa_g(q_{g,t}^{\text{sim}} - q_{g,t}^{\text{real}})), \quad (14)$$

$$\mathcal{L}_{\text{force}} = \frac{1}{3|\mathcal{T}_f|} \sum_{t \in \mathcal{T}_f} \sum_{k=1}^3 w_t v_{t,k} \text{Huber}_{\beta_f}(\hat{f}_{t,k} - f_{t,k}^{\text{gt}}). \quad (15)$$

For the OpenManipulator-X prismatic gripper, $\kappa_g = 1000$ converts the single-finger slide-coordinate residual from meters to millimeters before applying the loss. The force loss uses $\beta_f = 0.15$ for the force-sensor run and the final known-weight fine-tuning stage, and $\beta_f = 1$ otherwise. Franka instead supervises only the vertical component with $\mathcal{L}_{\text{force}}^{\text{Franka}} = |\mathcal{T}_f|^{-1} \sum_{t \in \mathcal{T}_f} w_t \text{Huber}_{0.15}(\hat{f}_{t,z} - f_{t,z}^{\text{gt}})$, where $f_{t,z}^{\text{gt}} = -mg$ is the known-payload label. The sample weight w_t implements fixed nonzero-force-sample reweighting. By default $v_{t,k} = 1$, including for missing-value sentinels converted to zero; configurations that enable invalid-channel masking set the corresponding $v_{t,k} = 0$. The classification terms are

$$\mathcal{L}_{\text{gate}} = \frac{1}{|\mathcal{T}_f|} \sum_{t \in \mathcal{T}_f} \text{BCE}(g_t, g_t^{\text{gt}}), \quad (16)$$

$$\mathcal{L}_{\text{cond}} = \frac{1}{|\mathcal{T}_c|} \sum_{t \in \mathcal{T}_c} \text{BCE}(c_{t,3}, m_{t,3}), \quad (17)$$

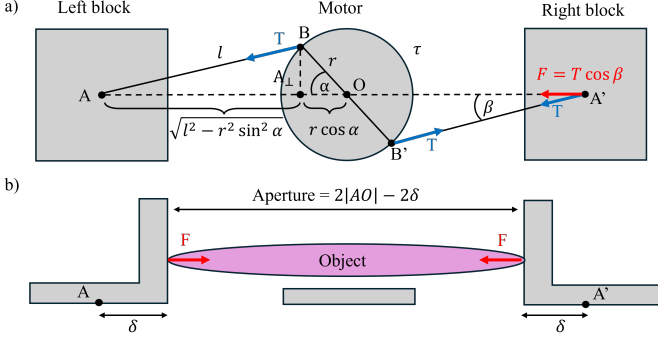


Fig. 6: **Kinematics and force transmission of the symmetric gripper.** Top (a) and side (b) views show the rotary motor driving two symmetric prismatic jaws. Each link transmits an axial force T , whose component along the closing direction gives the per-jaw force $F = T \cos \beta$.

where $m_{t,3} = 1$ denotes normal operation of Joint 3 and $m_{t,3} = 0$ denotes the mechanically restricted condition. For selected SO-101 and Franka runs, finite-difference velocities computed from logged poses provide an auxiliary Smooth L1 rollout loss $\mathcal{L}_{\text{vel}}$. The complete objective is

$$\mathcal{L} = \lambda_{\text{joint}} \mathcal{L}_{\text{joint}} + \lambda_{\text{grip}} \mathcal{L}_{\text{grip}} + \lambda_{\text{force}} \mathcal{L}_{\text{force}} + \lambda_{\text{gate}} \mathcal{L}_{\text{gate}} + \lambda_{\text{cond}} \mathcal{L}_{\text{cond}} + \lambda_{\text{vel}} \mathcal{L}_{\text{vel}}. \quad (18)$$

We use $\lambda_{\text{vel}} = 5$ for the velocity-supervised SO-101 runs and 5 or 10 across the Franka training stages; it is zero otherwise. Unavailable task losses are likewise assigned zero weight. For visual-supervision experiments, the silhouette objective is added to Eq. 18 and backpropagated through the path described above. We train with AdamW [34] for up to 100000 epochs; in practice the best checkpoints emerge within 30000 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 B uses an RTX 4090, and the hand-eye calibration measurements in Appendix C use an A6000. Network training and inference were implemented in JAX with JIT compilation [3] and GPU acceleration, and differentiable simulation for rigid-body dynamics was based on the MuJoCo engine and its MJX JAX implementation [45, 16]. Additional implementation details are provided in the Appendix.

Because the gripper is driven by a rotary motor while its fingers translate along prismatic joints, relating motor commands to jaw position requires a kinematic mapping. We describe this mechanism next.

E. Gripper Kinematics

Fig. 6 defines the physical gripper geometry. Let B and B' denote the two crank pins, $r = |OB| = |OB'|$ the crank radius, $l = |AB| = |A'B'|$ the link length, α the acute angle between either crank radius and the slider axis, and δ the offset from each slider pin (A or A') to the corresponding inner jaw

face. The symmetric jaw aperture is

$$a(\alpha) = 2|AO| - 2\delta = 2\left(\sqrt{l^2 - r^2 \sin^2 \alpha} + r \cos \alpha\right) - 2\delta. \quad (19)$$

Equation 19 provides the ideal motor-angle-to-jaw-aperture relation for the physical mechanism; the simulator represents the gripper with a calibrated single-finger prismatic coordinate. For a quasistatic symmetric grasp, the links act as two-force members carrying axial force T . If β is the angle between a link and the slider axis, each jaw applies the closing force $F = T \cos \beta$; torque balance about the motor axis gives $\tau = 2Tr \sin \gamma$, where $\gamma = \angle OBA = \angle OB'A'$. Thus, the gripper's force transmission depends on both motor torque and mechanism configuration.

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 reports rollout errors across horizons. The model remains accurate over short horizons for both revolute and prismatic joints, and the errors remain moderate over the evaluated longer horizons. Because these experiments replay the recorded current sequence, they evaluate recorded-current-conditioned trajectory propagation rather than counterfactual planning; model-predictive control would additionally require a model of the current response to candidate commands. The consistent trend across joints indicates that physics-in-the-loop learning captures effective history-dependent input-output behavior while retaining the structure of differentiable rigid-body dynamics. Force MAE is the componentwise mean over the evaluated trajectories, time steps, and three Cartesian output channels: $\text{MAE}_F(T) = (3NT)^{-1} \sum_{n,t,k} |f_{t,k}^{(n)} - \hat{f}_{t,k}^{(n)}|$. For payload-based benchmarks, the nominal reference is $\mathbf{f}^{\text{gt}} = [0, 0, -mg]^\top$, so the metric compares lateral predictions with zero and the vertical prediction with $-mg$. The directional force-sensor benchmark provides measured three-axis force references. For gauge pushing, the scalar gauge reading is assigned to the known contact axis and the orthogonal reference components are zero; the tabulated F values retain the same three-channel convention, whereas the trace-level values in Fig. 1 use the contact-axis force magnitude. For OpenManipulator-X, Grip denotes single-finger slide-coordinate MAE in millimeters, rather than the total jaw aperture $a(\alpha)$.

Across the rollout horizons $H \in \{64, 128, 256, 320, 500\}$ (Tab. II), simulator-input gradient norms remain $O(10^{-2})$, and parameter-gradient cosine similarity remains at least 0.96 relative to $H=128$; $\|\nabla_{\theta} \mathcal{L}\|$ plateaus around $H=256-320$, motivating the curriculum used in training.

TABLE II: **Gradient behavior across rollout horizons H .** Here, input-grad denotes the norm of the loss gradient at the simulator’s bounded generalized-effort input; cos is the parameter-gradient cosine similarity relative to $H = 128$.

H	input-grad	$\ \nabla_{\theta}\mathcal{L}\ $	cos
64	2.73×10^{-2}	8.77	0.96
128	2.59×10^{-2}	17.33	1.00
256	1.72×10^{-2}	22.35	0.99
320	1.45×10^{-2}	22.91	0.99
500	9.37×10^{-3}	20.29	0.98

TABLE III: **Simulation accuracy on the test set.** J1–J4: joint-angle MAE (deg). Grip: single-finger slide-coordinate MAE (mm). Rollouts of 600 steps span approximately 10 seconds at the platform sampling rate.

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

B. Force Estimation Accuracy

We evaluate force prediction on three complementary benchmarks: (i) directional contact measured by a calibrated six-axis force/torque sensor, (ii) known payloads whose gravitational loads define force labels, and (iii) contact with a force gauge across multiple directions and heights. The experimental setup is shown 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. On reference trajectories without contact, NeuralActuator predicts near-zero forces (0.00–0.02 N). Payloads at or below ~ 50 g (~ 0.5 N) lie close to the noise floor of this low-cost platform; reliable detection in this regime remains future work.

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

Unseen Contact Geometries.

The force head predicts base-frame end-effector force from the shared telemetry representation and does not take explicit object geometry as input. We therefore evaluate the pretrained model on two unseen objects (inset) whose shapes and surface properties are not represented in training: a 261 g payload and a 226 g payload. We measure



TABLE IV: **Simulation and force prediction accuracy on the force-sensor test set.** J1–J4: joint-angle MAE (deg). Grip: single-finger slide-coordinate 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
<i>With External Force Contact</i>																		
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
<i>Reference (No External Force)</i>																		
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

TABLE V: **Simulation and force prediction accuracy on the payload-based test set.** J1–J4: joint-angle MAE (deg). Grip: single-finger slide-coordinate 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	200 g	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
	300 g	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
	400 g	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	200 g	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
	300 g	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
	400 g	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
place	400 g	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
	500 g	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

the force during the stationary 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.

Qualitative Payload Rollouts. Figs. 7 and 8 show paired hardware executions and simulated rollouts for the two payload-based manipulation tasks. The force overlays capture contact transitions during pick-and-place and sustained loading during lift-and-hold, complementing the aggregate errors in Tab. V.

Gauge Pushing. We further test contact with a force gauge from the front (horizontal pushing) and top (vertical pressing) at low, middle, and high end-effector positions. Tab. VI reports state and force errors over 100-, 300-, and 600-step rollouts. Average force MAE increases from 0.08 N at 100 steps to 0.10 N at 600 steps and remains at most 0.12 N for every setting. Joint errors grow moderately with horizon, whereas single-finger slide-coordinate MAE remains approximately 0.21 mm. Fig. 9 shows that the predicted force also tracks contact onset, sustained loading, and release for representative front pushes.

C. Learned Torque Surrogates

Fig. 10 compares raw, pre-clipping torque-surrogate outputs with measured motor currents over time for the four revolute joints of OpenManipulator-X. Because this platform lacks joint-torque sensing, these curves serve as diagnostics of a multivariate, history-conditioned network rather than a calibrated current–torque relation. Clipping also makes the

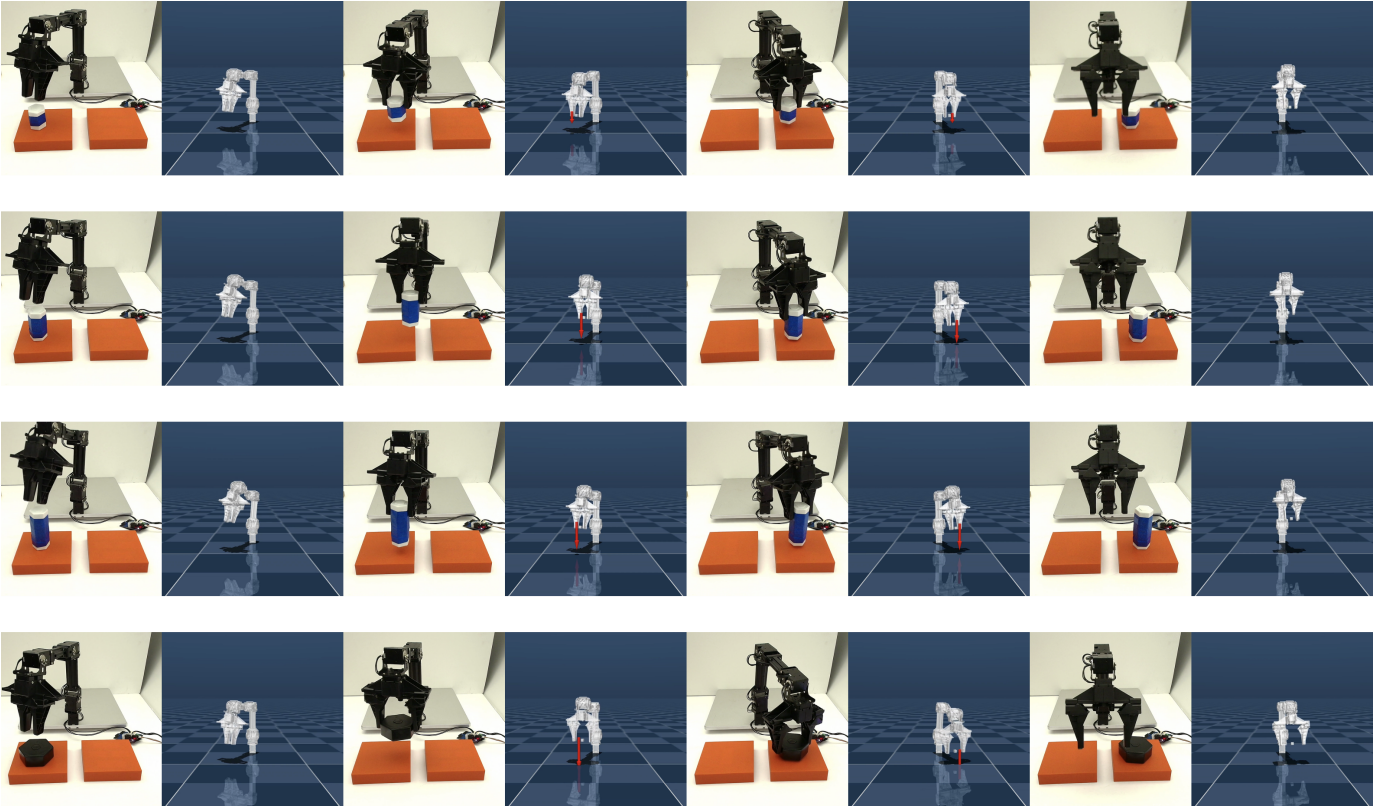


Fig. 7: **Force-aware pick-and-place rollouts.** From top to bottom: 200 g, 300 g, 400 g, and 500 g payloads. Each row pairs the real execution with the simulated rollout and overlays the predicted end-effector external force.

TABLE VI: **Simulation and force prediction accuracy on gauge pushing.** The end-effector contacts the gauge from the front or top at three positions. J1–J4: joint-angle MAE (deg). Grip: single-finger slide-coordinate MAE (mm). F: force MAE (N).

Task	@100 steps						@300 steps						@600 steps					
	J1	J2	J3	J4	Grip	F	J1	J2	J3	J4	Grip	F	J1	J2	J3	J4	Grip	F
high_push_front	0.91	1.68	1.18	0.62	0.00	0.07	1.62	1.29	0.53	0.32	0.00	0.08	1.48	1.16	1.44	1.09	0.00	0.09
high_push_top	0.73	1.01	0.78	1.61	0.01	0.10	1.38	1.35	1.97	1.92	0.01	0.11	1.62	1.57	2.72	2.01	0.01	0.12
mid_push_front	0.66	0.52	0.96	0.61	0.30	0.06	0.85	0.51	0.60	0.99	0.31	0.07	0.95	0.56	0.48	0.98	0.31	0.08
mid_push_top	0.49	0.54	0.87	0.74	0.30	0.09	0.64	0.43	0.73	1.07	0.31	0.10	0.89	0.52	0.75	0.78	0.31	0.11
low_push_front	0.61	0.75	1.83	0.23	0.32	0.08	0.87	0.56	1.65	0.33	0.32	0.09	1.29	0.88	1.12	0.98	0.33	0.10
low_push_top	1.00	0.36	1.05	0.29	0.31	0.08	1.17	0.84	2.11	0.74	0.32	0.09	1.35	0.76	2.27	1.04	0.32	0.10
Avg	0.73	0.81	1.11	0.68	0.21	0.08	1.09	0.83	1.27	0.90	0.21	0.09	1.26	0.91	1.46	1.15	0.21	0.10

pose loss insensitive to raw magnitudes outside the applied bounds; rollout-level interpretation therefore applies only to the bounded surrogate $\bar{\tau}$ supplied to the simulator in Eq. 10.

D. Comparison with Baselines

To compare NeuralActuator with model-based alternatives under a common low-cost-servo input protocol, we implement three adapted baselines on the payload benchmark: (i) *ID-Linear*, the acceleration-based inverse-dynamics residual $\hat{\tau}_{\text{ext}} = \tau_{\text{ID}}(q, \dot{q}, \ddot{q}) - (\mathbf{K}_t \dot{\mathbf{i}} + \mathbf{b})$ with per-joint $K_{t,j}$ calibrated on free motion; (ii) *ID-Friction*, which augments (i) with Stribeck friction and backlash terms; and (iii) the Generalized Momentum Observer (*GMO*) [10], which avoids $\ddot{q}$ through momentum integration with an auto-tuned gain.

The first two are simplified benchmark adaptations and differ from the camera-assisted contact-localization method of Margrini et al. [37] and the measured-motor-torque-based convex friction-uncertainty estimator of Linderoth et al. [31]. All three baselines convert motor current to torque through the linear current–torque map and recover end-effector force as $\hat{\mathbf{f}} = (J_v^\top)^\dagger \hat{\tau}_{\text{ext}}$; they consume *ground-truth* robot states at every step, whereas NeuralActuator predicts forces from its own simulated rollout. First, fitting K_t per joint on OpenManipulator-X reveals the limits of the linear assumption: two joints yield negative slopes before clamping, consistent with high-ratio gearboxes masking the underlying current–torque relation. Second, even after Stribeck and backlash compensation, the mean force MAE of the best adapted classical baseline (*GMO*) is 5.5 times that of NeuralActuator (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 versus 1.41 N), indicating that these fixed parametric models do not capture the dominant nonlinearities of the geared servos on this platform.

E. Online Adaptation

When actuator characteristics change across hardware instances or operating conditions, a pretrained model may require local recalibration. Starting from the pretrained checkpoint, we therefore fine-tune NeuralActuator in situ on 12

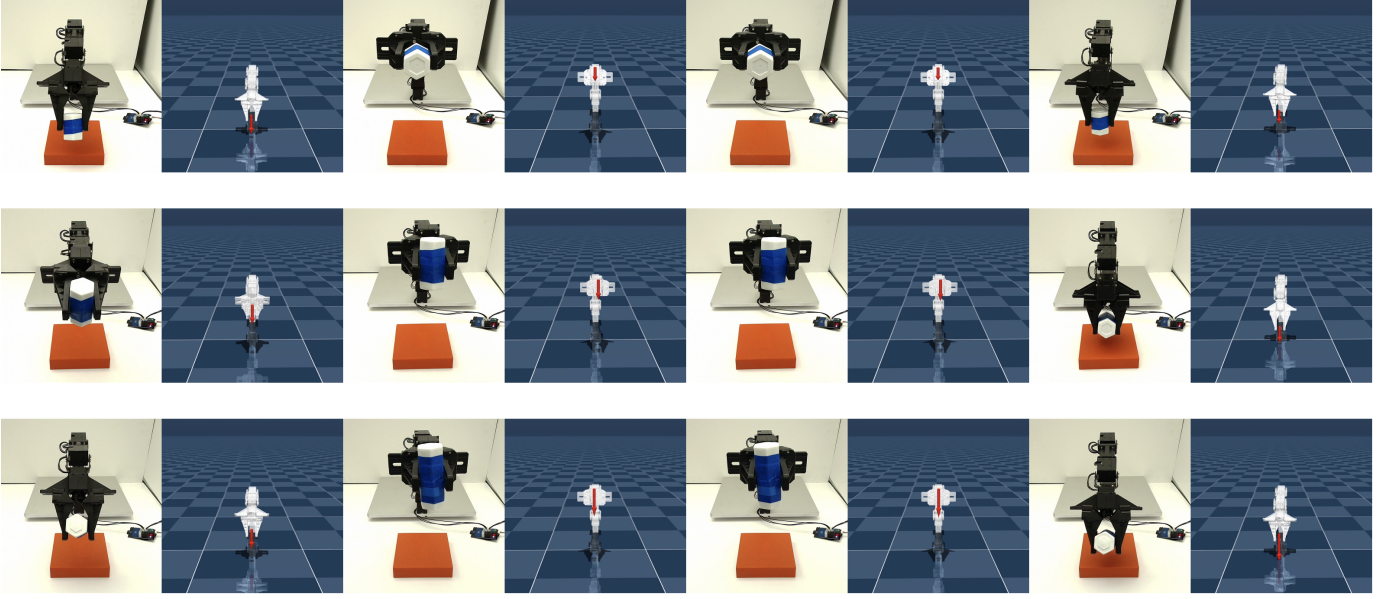


Fig. 8: **Force-aware lift-and-hold rollouts.** From top to bottom: 200 g, 300 g, and 400 g payloads. Each row pairs the real execution with the simulated rollout and overlays the predicted end-effector external force.

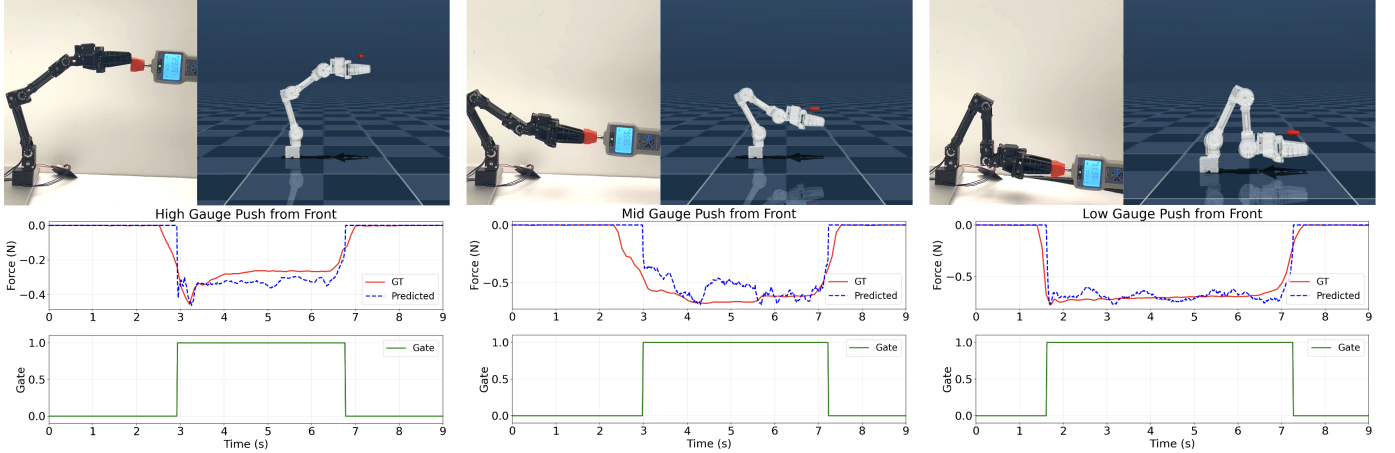


Fig. 9: **Force prediction for front-direction gauge pushing.** For high, middle, and low contact positions, we show the physical setup and simulated scene together with the measured and predicted signed contact-axis force. The bottom panels show the corresponding binary contact interval.

TABLE VII: **Force MAE (N) against model-based sensorless baselines** on the payload benchmark at a 100-step horizon (lower is better). Baselines consume ground-truth states at every step; NeuralActor uses simulated rollout states.

Method	Go Up & Stay			Pick & Place				Avg
	200 g	300 g	400 g	200 g	300 g	400 g	500 g	
ID-Linear	1.37	1.81	2.30	0.72	0.95	1.22	1.47	1.41
ID-Friction	1.06	1.59	2.15	0.62	0.82	1.10	1.31	1.23
GMO [10]	0.58	0.66	1.23	0.33	0.47	0.63	0.75	0.66
NeuralActor	0.12	0.20	0.07	0.24	0.00	0.19	0.05	0.12

newly collected trajectories (7,810 frames). The adaptation runs for 70 epochs and takes 356 seconds end-to-end, corresponding to approximately 5.1 seconds per epoch.

Tab. VIII reports per-joint position, single-finger slide-coordinate, and force errors before (B) and after (A) adaptation

on the force-sensor benchmark. Adaptation reduces aggregate position-tracking error across all three rollout horizons, while force MAE remains comparable and improves slightly at the longer horizons. Fig. 11 reports approximately 44% average tracking-error reduction after 70 epochs under the curve-level aggregation used for that progress plot; this statistic is distinct from the horizon-wise Avg rows in Tab. VIII. In this setting, recalibration required 12 trajectories and 356 seconds rather than full retraining.

F. Cross-Platform Validation

For clarity, OpenManipulator-X has four arm joints plus one gripper actuator, SO-101 has five arm joints plus one gripper actuator, and Franka has seven arm joints. The Franka results below report external force only; the SO-101 table reports its arm joints and treats gripper actuation separately.

TABLE VIII: **Simulation and force prediction accuracy before (B) and after (A) online adaptation.** J1–J4: joint-angle MAE (deg). Grip: single-finger slide-coordinate 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	
	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A		
With External Force Contact																																				
force_X+	0.9	0.3	2.1	0.7	1.0	0.8	0.8	0.8	1.0	1.0	0.36	0.37	2.1	1.2	2.3	1.0	0.7	1.5	0.9	0.8	1.0	1.0	0.46	0.41	2.7	1.6	4.0	1.3	0.6	1.9	0.9	0.8	1.0	1.0	0.52	0.41
force_X-	0.2	0.5	1.1	1.7	0.9	1.2	0.7	1.3	1.0	1.0	0.39	0.39	1.7	0.7	1.6	1.6	1.4	0.7	0.8	1.2	1.0	1.0	0.41	0.40	2.5	0.9	1.9	1.6	2.0	0.8	0.9	1.2	1.0	1.0	0.42	0.41
force_Y+	2.1	1.8	5.4	3.4	0.8	1.1	1.2	1.3	0.0	0.1	0.37	0.37	2.3	1.5	4.5	3.2	1.1	1.1	1.6	1.0	0.1	0.1	0.39	0.37	3.0	1.2	3.7	2.3	2.5	1.2	3.1	1.7	0.3	0.3	0.45	0.39
force_Y-	1.1	1.2	0.6	1.3	1.1	1.8	1.8	1.8	0.1	0.1	0.38	0.38	1.6	1.0	3.1	2.6	1.2	1.4	2.1	2.0	0.1	0.1	0.36	0.36	1.7	1.0	2.3	2.0	2.5	1.1	2.1	1.6	0.2	0.2	0.38	0.37
force_Z+	2.9	4.0	5.2	3.6	1.3	3.1	0.6	0.5	0.1	0.1	0.39	0.39	2.7	2.1	4.0	2.7	1.4	1.5	0.7	0.9	0.2	0.1	0.44	0.42	5.0	1.8	3.2	2.0	1.7	1.5	1.4	1.5	0.3	0.2	0.46	0.45
force_Z-	0.7	0.9	6.2	3.5	2.0	0.6	1.0	0.7	1.0	1.0	0.36	0.36	2.0	0.7	7.1	2.3	1.8	0.8	3.3	1.2	1.0	1.0	0.61	0.38	2.6	1.0	7.2	1.6	2.0	0.8	3.0	1.1	1.0	1.0	0.54	0.40
Reference (No External Force)																																				
force_X+	1.4	0.4	1.6	0.4	1.0	1.0	0.8	0.8	1.0	1.0	0.01	0.08	1.5	0.5	1.3	0.6	0.8	0.8	1.0	1.2	1.0	1.0	0.02	0.08	1.3	0.5	1.1	0.6	0.6	1.0	1.4	1.9	1.0	1.0	0.01	0.05
force_X-	0.4	0.7	3.2	0.6	2.4	0.5	0.7	1.0	1.0	1.0	0.00	0.00	0.7	0.6	2.4	1.2	1.9	0.5	0.9	0.9	1.0	1.0	0.00	0.04	0.8	0.6	2.2	1.0	2.2	0.7	1.1	0.8	1.0	1.0	0.00	0.05
force_Y+	1.4	1.8	1.0	2.8	1.4	1.4	1.3	1.9	0.1	0.1	0.00	0.00	2.2	1.7	3.3	3.3	1.0	1.2	1.8	1.4	0.2	0.2	0.00	0.01	2.2	1.2	4.4	2.4	1.0	1.1	1.3	1.3	0.4	0.3	0.00	0.01
force_Y-	2.0	1.2	2.2	1.9	0.8	0.4	2.7	1.3	0.0	0.1	0.00	0.01	2.5	1.5	4.8	2.7	0.7	0.8	3.2	1.8	0.1	0.1	0.00	0.01	2.0	1.2	4.9	2.1	0.7	1.3	2.7	2.0	0.3	0.2	0.00	0.01
force_Z+	0.6	0.9	1.3	0.8	0.4	1.3	1.4	1.3	0.1	0.1	0.00	0.00	1.3	1.2	3.4	1.6	0.4	1.0	1.2	1.0	0.2	0.2	0.00	0.00	1.0	1.1	3.4	1.5	0.5	1.2	0.9	1.3	0.3	0.3	0.00	0.00
force_Z-	1.8	0.5	1.9	0.8	1.1	0.9	0.5	0.7	1.0	1.0	0.03	0.14	1.7	0.6	0.9	0.7	1.1	0.8	0.8	0.8	1.0	1.0	0.02	0.11	1.6	0.5	0.8	0.7	1.0	0.7	0.8	1.1	1.0	1.0	0.01	0.08
Avg	1.28	1.19	2.65	1.81	1.19	1.17	1.13	1.12	0.55	0.55	0.19	0.21	1.86	1.09	3.22	1.96	1.11	1.02	1.53	1.18	0.59	0.58	0.23	0.22	2.20	1.04	3.25	1.58	1.44	1.10	1.64	1.35	0.66	0.65	0.23	0.22

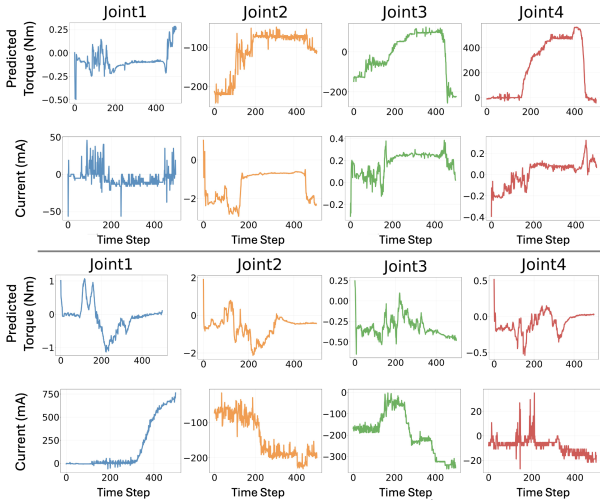


Fig. 10: **Measured motor current and raw torque-surrogate output for the arm joints.** Two representative rollouts compare current with the pre-clipping network output. The bounded surrogate in Eq. 10 enters the simulator; the raw curves are diagnostic outputs rather than direct motor-torque measurements. At a 500-step rollout horizon, the average joint-angle and single-finger slide-coordinate MAEs are 1.26° and 1 mm, respectively.

7-DoF Franka Emika Panda: external-force estimation.

The Franka model is command-conditioned: its input contains a target joint pose together with the current state, commanded joint torque, motor-side state, and gripper features. Because the logged commanded-position signal is a sparse sequence of setpoints, the offline feature construction replaces that channel with the smooth proxy $\tilde{\mathbf{q}}_t^{\text{cmd}} = 1.03 \mathbf{q}_{t+5}^{\text{rec}}$, constructed from the recorded joint positions. Because this proxy depends on a future recorded state, the reported Franka force results constitute a future-state-conditioned offline benchmark rather than an online evaluation. For causal hardware deployment, this channel must instead be populated by the commanded pose already available at time t from the controller, teleoperation interface, or inverse-kinematics module; no future measured state is then required. Accordingly, the Franka analysis focuses

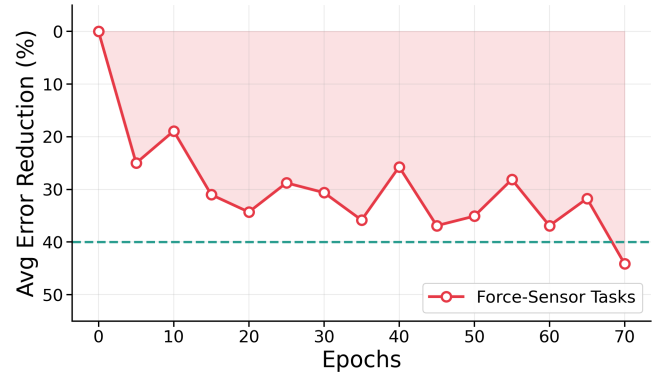


Fig. 11: **Online adaptation progress.** The curve-level aggregate shows approximately 44% average tracking-error reduction after 70 epochs with 12 trajectories (7,810 frames). Fine-tuning takes 356 seconds in total (approximately 29.7 seconds per trajectory and 5.1 seconds per epoch).

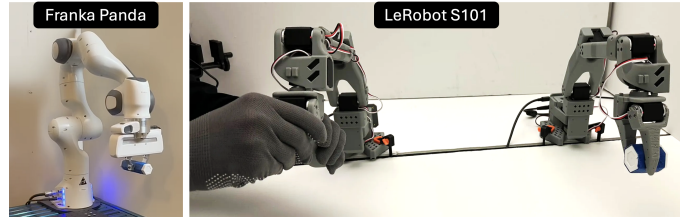


Fig. 12: **Cross-platform validation.** We train NeuralActuator from scratch on a 7-DoF Franka Emika Panda and a 6-DoF SO-101 low-cost arm with STS3215 servos. Franka provides an offline payload-force-estimation benchmark, whereas SO-101 provides both rollout and force evaluation. Together with OpenManipulator-X, these robots span three actuator families and costs from approximately \$500 to more than \$30,000.

on external-force estimation. The force head is supervised by the known-payload vertical load and operates in parallel with the simulator-control head on the shared Transformer representation. Across 200–600 g lift-and-hold trajectories, the reported three-component force MAE ranges from 0.26 to

TABLE IX: **Franka Panda external-force output MAE (N)** at 100- and 500-step future-state-conditioned offline rollouts. F averages the three output components against the nominal payload reference $[0, 0, -mg]^T$. Consistent with the vertical-loading protocol, training supervision is applied to f_z , while the lateral outputs enter the reported metric through their zero references.

Payload	F @100	F @500
200 g	0.42	0.31
300 g	0.35	0.28
400 g	0.36	0.27
500 g	0.27	0.26
600 g	0.30	0.28
Avg	0.34	0.28

TABLE X: **SO-101 per-joint and force MAE** at a 500-step rollout. Joint errors are in degrees and force error (F) is in N.

Task	Go Up & Stay						Pick & Place					
	J1	J2	J3	J4	J5	F	J1	J2	J3	J4	J5	F
300 g	2.27	8.20	6.40	5.40	2.37	0.64	3.73	5.63	4.17	6.17	2.47	0.73
400 g	2.83	9.30	9.63	4.00	5.73	0.57	3.53	6.83	2.43	5.23	5.13	0.63
500 g	2.27	7.63	9.77	7.13	2.67	0.47	3.80	5.60	1.53	6.73	3.33	0.54

0.42 N. Tab. IX reports the payload-wise results at 100- and 500-step horizons.

6-DoF low-cost arm (SO-101). SO-101 uses STS3215 serial-bus servos, extending the cross-platform evaluation beyond the DYNAMIXEL family used in the primary NAD experiments. Its model uses the servos’ signed load registers as the effort-related input; the raw current registers are not used. The complete SO-101 corpus contains 65,916 frames (1,037.7 s at an aggregate rate of approximately 63.5 Hz) across ten combinations of two tasks and five payload conditions. For a controlled comparison across nonzero payloads, the evaluation subset contains 40,426 frames (649 s, approximately 62.3 Hz) across six combinations: go-up-and-stay and pick-and-place with 300 g, 400 g, and 500 g payloads. At a 500-step horizon, force MAE is 0.47–0.73 N, and the largest joint error is approximately 9.8° on Joint 3 (Tab. X). The larger joint errors relative to OpenManipulator-X may reflect differences in actuator and platform dynamics together with the finite per-platform training set; the short calibration procedure in Sec. IV-E provides a practical mechanism for platform-specific refinement.

G. Motor Condition Estimation

Next, we create a controlled mechanical restriction by wrapping rubber bands around Joint 3 to introduce additional resistance (Fig. 13(a)). Under identical position commands, the restricted joint draws more current while following a similar trajectory (Fig. 13(b)). This current deviation enables sensorless classification of the restricted condition. For evaluation, the restricted-class score is $1 - c_3$. Under the native decision granularity of each method, NeuralActuator achieves 91.0% accuracy, 96.2% recall, and 0.95 AUC-ROC on a pick-

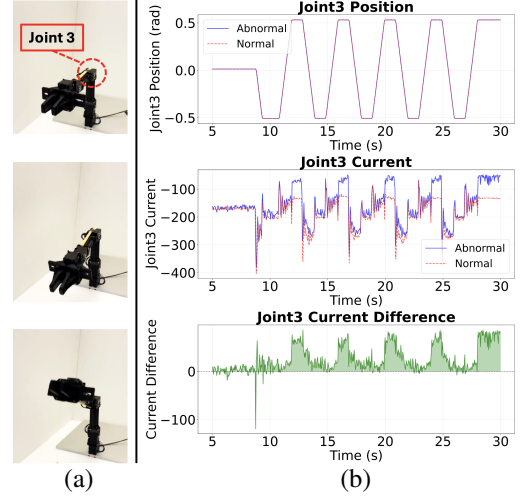


Fig. 13: **Motor-condition experiment.** (a) Rubber bands mechanically restrict Joint 3. (b) Under matched commands, the unrestricted and restricted trials follow similar position trajectories (top), but the restricted joint draws more current (middle), producing a measurable difference (bottom).

and-place task with a 200 g object (Tab. XI), with higher reported values than the threshold, SVM, and Random Forest baselines. The reported task distinguishes unrestricted from mechanically restricted operation and does not address motor-damage diagnosis or general motor-health assessment.

TABLE XI: **Motor-condition classification under controlled mechanical restriction.** Precision and recall treat the mechanically restricted condition as the positive class. The handcrafted baselines classify overlapping 64-step windows, whereas NeuralActuator uses a nine-frame context and produces framewise predictions. All methods are evaluated on the same four held-out trajectories.

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

Baseline Protocols. The threshold baseline classifies a window using the mean absolute Joint 3 current, with the threshold selected on the training set. The SVM uses an RBF kernel, and the Random Forest uses 200 trees with maximum depth 15. Both operate on the same 20 handcrafted features: six current statistics (mean, standard deviation, maximum, minimum, skewness, and kurtosis); three tracking-error statistics (mean, maximum, and standard deviation); four temporal features (mean and maximum current derivative, mean acceleration, and current trend); three spectral features (dominant-frequency power, low-to-high-frequency power ratio, and spectral entropy); three physics-inspired features (current during motion, current-to-velocity ratio, and power estimate); and one first-half-versus-second-half current-asymmetry feature. NeuralActuator is trained on 32 trajectories (eight from each task–condition combination), whereas the three handcrafted

TABLE XII: Model parameters and runtime performance.

Model Sizes			Inference		
Metric	Value	Unit	Metric	Value	Unit
Parameters	1.44M	–	Mean time	0.25	ms
FLOPs (fwd)	5.46M	–	P95 time	0.31	ms
FP32 parameter memory	5.50	MiB	Thru. (b=1)	4,019	Hz
			Thru. (b=32)	10,992	Hz

baselines are fitted on 16 trajectories (four from each combination). All methods are evaluated on the same four held-out trajectories (two per condition). They classify 64-step windows with 50% overlap, yielding 601 training and 152 test windows; the SVM and Random Forest features are standardized using training-set statistics only. NeuralActuator instead produces a prediction at each time step from its native nine-frame input (eight historical frames plus the current frame) and is evaluated on the same four held-out test trajectories. Metrics are therefore reported at each method’s native decision granularity, with the same four held-out trajectories providing the common evaluation set.

H. Runtime Performance

Tab. XII shows that JAX with JIT compilation yields sub-millisecond GPU inference latency and throughput above the 60 Hz control rate. Batch processing further improves throughput, and the parameter footprint remains modest.

I. Force-Aware Imitation Learning for Real-Robot Control

To evaluate the utility of NeuralActuator as a pretrained force-perception module for downstream control, we train a behavior-cloning (BC) controller using a frozen NeuralActuator module. Following [29], we collect expert demonstrations via teleoperation and train a policy network to predict commanded joint positions (see details in Appendix F). 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. The pretrained NeuralActuator module is trained on the full 100–500 g payload set, while the BC demonstrations include the 500 g condition used in the pick-and-place evaluation. The downstream protocol therefore evaluates task-level transfer within the payload range represented during pretraining. **Force-aware policy input.** The BC policy receives a history of joint positions and gripper aperture augmented with the **external force signal** predicted by the frozen NeuralActuator module, $\hat{\mathbf{f}}_{\text{ext}}$. This augments position-only control with force feedback under varying payloads and intermittent contacts. The NeuralActuator module remains frozen throughout policy training; it is pretrained on teleoperated NAD trajectories and is not optimized jointly with the downstream policy. **Position-only baseline.** The baseline trains the same policy architecture from the position-and-aperture history without the predicted-force input. Tab. XIII shows higher success rates for the force-aware variant than for the position-only baseline in the evaluated manipulation trials (Fig. 1).

TABLE XIII: 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%

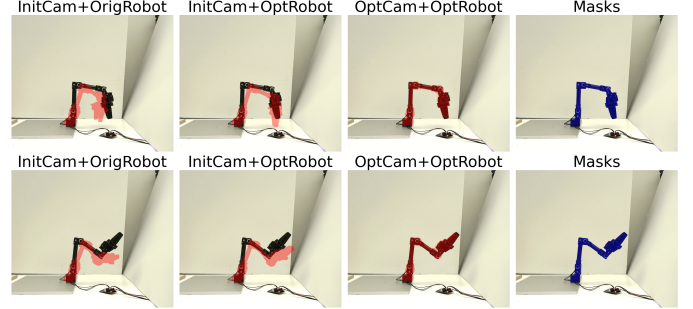


Fig. 14: **Image-space refinement.** The evaluated pipeline refines the hand–eye transform and NeuralActuator parameters through the simulated robot state; the visualization reports silhouette alignment.

J. Visual Supervision

We evaluate an image-space refinement pipeline that composes the neural actuator, differentiable dynamics, forward kinematics, and a differentiable renderer. A geometric three-point initialization provides the camera-from-base transform T_{cb} , after which the silhouette objective refines the camera extrinsics and NeuralActuator parameters for the evaluated sequence. Camera intrinsics are estimated with VGGT [46] and held fixed; SAM3 produces the observed robot masks [5]. Following the EasyHeC formulation [6], gradients propagate from mask alignment through the renderer and simulated states to NeuralActuator. The resulting silhouette alignment reaches a mean IoU of 0.8515 (Fig. 14); this experiment evaluates alignment quality and does not by itself quantify improvements in actuator or rollout accuracy. Appendix C gives the complete objective and optimization details.

K. Ablation on Network Architecture

To isolate the effect of network architecture, we compare our Transformer against three alternatives with approximately matched parameter counts (~ 1.4 M). Let $\mathbf{x}_t \in \mathbb{R}^F$ denote the feature vector containing joint states, motor currents, and goals. We consider: (i) an MLP with LayerNorm that predicts the torque surrogate from a flattened history, $\tau_t = f_\theta(\mathbf{x}_{t-L+1:t})$; (ii) a GRU [8], $\mathbf{h}_t = \text{GRU}_\theta(\mathbf{x}_t, \mathbf{h}_{t-1})$, with a learnable initial state; and (iii) an LSTM [21], $(\mathbf{h}_t, \mathbf{c}_t) = \text{LSTM}_\theta(\mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1})$. At a 500-step horizon, the Transformer is best or tied for best on four of the six reported metrics, including force MAE (Tab. XIV).

L. Additional Physics Backend

The learned actuator can also be coupled to differentiable rigid-body simulators other than MuJoCo. We integrate NeuralActuator with NVIDIA Warp [36] through a lightweight

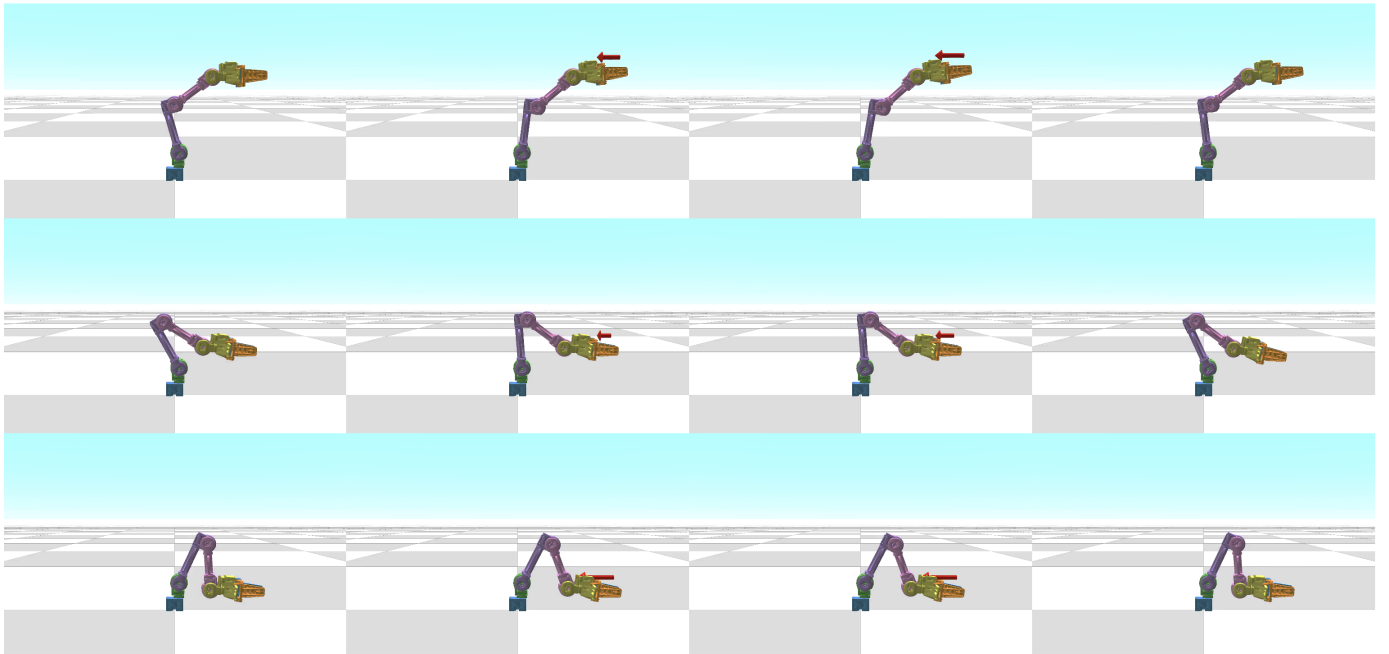


Fig. 15: **NeuralActuator with NVIDIA Warp.** Representative articulated-arm rollouts under different initial configurations and external push directions (red arrows), simulated with `wp.sim.FeatherstoneIntegrator`.

TABLE XIV: **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

adapter that maps generalized joint states and control signals between the actuator model and Warp’s articulated-body simulator. The implementation advances the robot with `wp.sim.FeatherstoneIntegrator` and records each rollout with `wp.Tape()` for reverse-mode differentiation. We use a gravitational acceleration of -9.81 m/s^2 , an environment rate of 60 Hz, five simulation substeps per frame (300 Hz internally), and a per-joint armature value of 0.01. Fig. 15 shows representative differentiated rollouts under multiple initial configurations and external pushes.

V. CONCLUSION

We presented NeuralActuator, a differentiable neural actuator model for low-cost robots that supports torque-surrogate prediction and contact-gated force perception without dedicated force/torque sensors at inference time. Instead of assuming a fixed linear current–torque relation, NeuralActuator learns a history-dependent mapping from command, state, and telemetry histories to a simulator-equivalent generalized input, while jointly estimating external forces and contact.

We provide a twin-arm teleoperation pipeline for collecting force-labeled data and a multi-task Transformer that predicts torque surrogates, forces, contact probabilities, and motor-condition scores; its torque-surrogate head is trained through differentiable simulation without direct generalized-effort labels. Experiments across three platforms spanning three actuator families and costs from approximately \$500 to more than \$30,000 show accurate motion prediction, force estimation on payload benchmarks, Joint 3 condition estimation, and improved behavior-cloning performance with NeuralActuator. *Limitations and Future Work.* Long-horizon recorded-telemetry-conditioned trajectory propagation accumulates error and eventually degrades tracking accuracy; the few-trajectory online adaptation routine in Sec. IV-E mitigates this drift. On pose-supervised platforms, the learned torque surrogate is simulator-equivalent: when contacts or payloads are absent from the forward model, the surrogate may absorb their generalized effects as well as other dynamics mismatch, while the force head remains a separate supervised estimator. Online inference also requires live effort-related actuator telemetry. Although causal closed-loop deployment is supported as new telemetry arrives, the current model does not predict the effort response to unexecuted commands and therefore cannot perform counterfactual rollouts over candidate future command sequences without an additional effort-response model. The force head estimates a single 3D resultant force at the end effector; it neither localizes distributed or multipoint contacts nor estimates a full wrench, and it requires explicit force labels for direct supervision. Future work could explore multipoint force prediction and more scalable ways to learn force estimation from visual cues.

VI. ACKNOWLEDGMENTS

This work was conducted as part of the MIT-Amazon Science Hub. We thank Yuxiang Ma and Yuxin Song for assistance with the hardware experiments. We thank the anonymous reviewers for their constructive feedback.

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] 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.
- [3] 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>.
- [4] 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.
- [5] 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>.
- [6] 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>.
- [7] 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.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] 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.
- [13] 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>.
- [14] Wanjia Fu, Hongyu Li, Ivy X He, Stefanie Tellex, and Srinath Sridhar. UniTac: Whole-robot touch sensing without tactile sensors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 3976–3985, June 2026.
- [15] Genesis Authors. Genesis: A generative and universal physics engine for robotics and beyond, December 2024. URL <https://github.com/Genesis-Embodied-AI/Genesis>.
- [16] Google DeepMind. MuJoCo XLA (MJX). <https://mujoco.readthedocs.io/en/stable/mjx.html>, 2026. Official software documentation, accessed July 13, 2026.
- [17] 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.
- [18] 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, 2022. doi: 10.1109/TIE.2021.3095820.
- [19] 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.
- [20] 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.
 - [21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
 - [22] Taylor A Howell, Simon Le Cleac’h, Jan Brüdigam, J Zico Kolter, Mac Schwager, and Zachary Manchester. Dojo: A differentiable physics engine for robotics. *arXiv preprint arXiv:2203.00806*, 2022.
 - [23] 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, 2018. doi: 10.1109/TIE.2017.2748056.
 - [24] 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.
 - [25] 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.
 - [26] Maged Iskandar, Alin Albu-Schäffer, and Alexander Dietrich. Intrinsic sense of touch for intuitive physical human-robot interaction. *Science Robotics*, 9(93):eadn4008, 2024.
 - [27] 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.
 - [28] 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.
 - [29] Masato Kobayashi, Thanpimon Buamanee, Yuki Urinishi, and Haruo Takemura. ILBiT: Imitation learning for robot using position and torque information based on bilateral control with transformer. *IEEE 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*, 9(9):8170–8177, 2024. doi: 10.1109/LRA.2024.3428909.
 - [31] Magnus Linderöth, 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] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.
 - [35] Tribotix Pty Ltd. PincherX 150 5DOF robot arm. <https://tribotix.com/product/pincherx-150-robot-arm/>, 2020. Accessed: 2025-08-12.
 - [36] 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).
 - [37] 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.
 - [38] 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.
 - [39] 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.
 - [40] 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.
 - [41] ROBOTIS Co., Ltd. OpenMANIPULATOR-X: Overview (RM-X52-TNM). https://emmanual.robotis.com/docs/en/platform/openmanipulator_x/overview/, 2017. Accessed: 2025-07-12.
 - [42] 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.
 - [43] 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.
 - [44] 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.

- [45] 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.
- [46] 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.
- [47] 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.
- [48] 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.
- [49] Jie Xu, Eric Heiden, Ireteyayo Akinola, Dieter Fox, Miles Macklin, and Yashraj Narang. Neural robot dynamics. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 3915–3935. PMLR, 2025. URL <https://proceedings.mlr.press/v305/xu25g.html>.
- [50] 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.
- [51] 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.
- [52] 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 *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 652–669. PMLR, 2025.

APPENDIX

This appendix provides dataset details (Sec. A), implementation details (Sec. B), the visual-supervision formulation (Sec. C), an architecture comparison (Sec. D), a force-sensing taxonomy (Sec. E), and the imitation-learning setup (Sec. F).

A. Additional Details of the Neural Actuation Dataset (NAD)

a) Task Design: We collect the Neural Actuation Dataset (NAD) with a twin-arm leader-follower teleoperation system to capture actuator behavior across a broad range of motions, loads, and contact wrenches. Tab. A15 summarizes the task suite, which is organized into three categories. **(i) Free motion** tasks contain no external contact and are used to characterize nominal actuation dynamics under representative workspace motions, including end-effector circular trajectories (clockwise/counterclockwise), per-channel sweeps (actuator channels 1–5), and scripted motion primitives (lean-back/extend-forward, pick-and-place without payload, and go-up/hold-still). **(ii) Force-labeled** tasks introduce controlled external wrenches in two ways: (a) *payload* variants of go-up/hold-still and pick-and-place with task-specific payloads spanning 100–500 g to induce repeatable gravity loading, and (b) *directional interaction* trials in $\pm X$, $\pm Y$, and $\pm Z$ using an external fixture instrumented with a six-axis F/T sensor. For each directional trial, we additionally record a matched *no-interaction* counterpart that executes the same commanded motion but without physical contact, enabling paired characterization of contact-induced effects. **(iii) Motor-condition** tasks replicate pick-and-place sequences under nominal and mechanically restricted operation. We randomly split the 10 trajectories for each task instance into training, validation, and test sets in an 8:1:1 ratio.

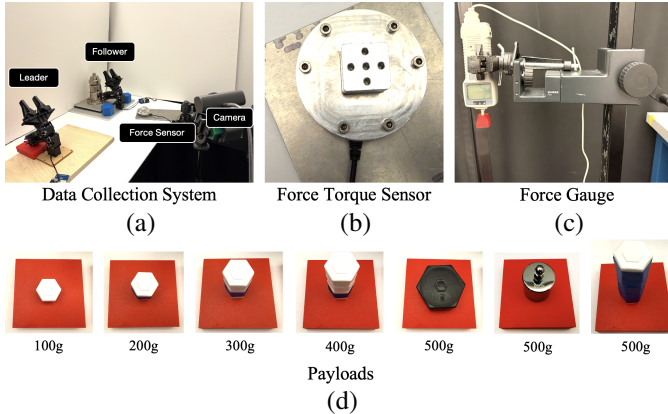


Fig. A16: **Data collection hardware.** (a) Twin-arm leader-follower teleoperation setup with a fixed external RGB camera and a fixture-mounted six-axis force/torque (F/T) sensor. (b) Close-up of the F/T sensor mounted on the rigid fixture. (c) Force gauge with a stand. (d) Payload set (100 g, 200 g, 300 g, 400 g, 500 g) used to induce controlled loading during data collection.

b) Data Capture Setup: Fig. A16 illustrates the collection apparatus. We perform teleoperation in a leader-follower manner: a human operator drives the leader arm, while the follower arm tracks the leader’s motion to execute the corresponding interaction in the physical scene (Fig. A16(a)). To obtain force supervision, the target object is mounted on a rigid fixture instrumented with a calibrated six-axis F/T sensor (Fig. A16(b)), which measures the interaction wrench $(F_x, F_y, F_z, \tau_x, \tau_y, \tau_z)$. We use only the three translational force channels as learning targets and express them in the robot base frame. For payload-based trials, we additionally attach standardized weights (100–500 g; Fig. A16(d)) to induce repeatable gravity loading. A fixed external RGB camera observes the workspace and records synchronized visual observations. All streams (robot proprioception, F/T readings, and camera frames) are timestamped and time-aligned per trajectory. For reproducibility, we (i) zero the F/T sensor prior to each recording session and after any re-mounting, (ii) keep the camera pose and the sensor-object fixture unchanged throughout a session, and (iii) use a consistent coordinate convention across trials, defining the $\pm X/\pm Y/\pm Z$ interaction directions in the F/T sensor frame.

c) Robotic Arm: We use two OpenManipulator-X arms with identical kinematic structures to form the leader and follower. Each arm is a 5-DoF platform (4 revolute joints + a 1-DoF parallel gripper) driven by DYNAMIXEL XM430-W350-T motors². The gear ratio of this motor is 353.5 : 1. The system is powered at 12 V and communicates over a TTL-level multidrop bus; the arms are controlled from a PC via an OpenCR interface. Each arm has a reach of 380 mm, a payload capacity of 500 g, repeatability below 0.2 mm, a maximum joint speed of 46 RPM, and a gripper stroke of 20–75 mm. Using two identical arms reduces kinematic mismatch between demonstration and execution and improves the repeatability of the collected trajectories.

Payload-labeled manipulation (SO-101). Tab. A16 summarizes the SO-101 collection. Following the same protocol as the OpenManipulator-X setup described in the main paper, we collect teleoperated trajectories under known payload conditions ($m \in \{0, 200, 300, 400, 500\}$ g) with two tasks: *go up and stay still* and *pick and place*. For each of the 10 task-payload combinations, we record 10 trajectories: eight for training, one for validation, and one for testing. The LeRobot logs contain 65,916 frames over 1,037.7 s, corresponding to an aggregate rate of approximately 63.5 Hz.

The task breakdown in Tab. A15 contains 350 OpenManipulator-X trajectory assignments across 35 rows. The two nominal motor-condition rows reuse 20 trajectories already listed under free motion and force-labeled interaction, yielding 330 distinct OpenManipulator-X trajectories. Together with 100 trajectories spanning ten SO-

²<https://emmanual.robotis.com/docs/en/dxl/x/xm430-w350/>

TABLE A15: **Task Composition of the Dataset.** The dataset consists of free-motion, force-labeled, and motor-condition data. Each task instance contains 10 trajectories, which are randomly split into 8 training, 1 validation, and 1 test trajectory. The two nominal motor-condition rows reuse trajectories listed in the free-motion and force-labeled categories.

Category	Task		#Frames	Duration (s)
Free motion (No external force)	Circular trajectory	Clockwise	8615	147.06
		Counterclockwise	8428	143.90
	Joint sweep	Motor 1	22935	392.04
		Motor 2	8688	148.47
		Motor 3	11357	193.95
		Motor 4	15012	256.57
		Motor 5	7338	125.21
	Lean back and extend forward		12261	209.35
	Pick & place (empty)		15288	261.08
	Go up and stay still		10011	171.14
Force-labeled	Go up and stay still (with weight)	100 g	10976	187.61
		200 g	10606	181.26
		300 g	11259	192.40
		400 g	11291	192.89
	Pick and place (with weight)	100 g	12483	213.35
		200 g	13245	226.15
		300 g	13129	224.31
		400 g	15124	258.54
		500 g	13957	238.49
	Force sensor	+X	5005	85.17
		+X w/o interaction	5005	85.21
		−X	5169	87.94
		−X w/o interaction	5169	88.00
		+Y	7444	126.74
		+Y w/o interaction	7444	126.78
		−Y	6921	117.88
		−Y w/o interaction	6921	117.91
		+Z	6906	117.59
		+Z w/o interaction	6906	117.63
		−Z	5565	94.69
		−Z w/o interaction	5565	94.73
Motor condition	Pick & place w/ weight (200 g)	Motor restricted	10113	168.26
		Nominal motor	13245	226.15
	Pick & place (empty)	Motor restricted	11252	192.41
		Nominal motor	15288	261.08

Category	Task		#Frames	Duration (s)
Payload-labeled	Go up and stay still	Empty	6753	87.5
		200 g	7312	117.4
		300 g	7227	116.1
		400 g	7659	123.0
		500 g	7630	122.5
	Pick and place	Empty	6125	98.4
		200 g	5300	85.1
		300 g	5553	89.2
		400 g	5725	92.0
		500 g	6632	106.5

TABLE A16: **SO-101 dataset overview.**

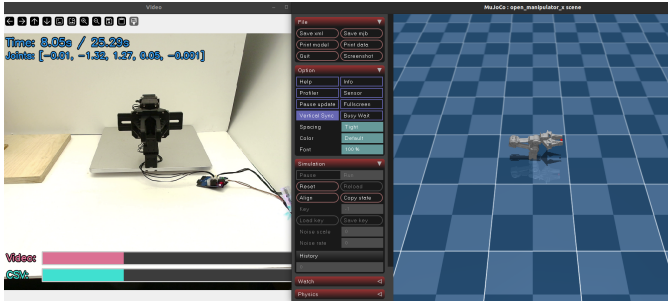
101 task–payload combinations, the complete NAD contains 430 distinct trajectories. Individual training and evaluation

protocols use task-specific subsets of this broader collection, and collection videos are available for all 430 trajectories.

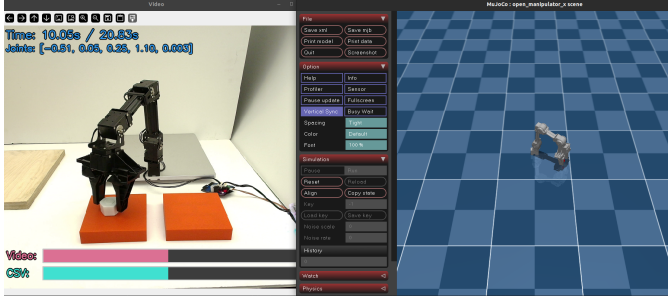
B. Implementation Details

a) *Loss Functions:* We use robust regression losses for continuous targets and binary cross-entropy for classification targets. Using implementation-specific weights, Eq. 18 can be written as

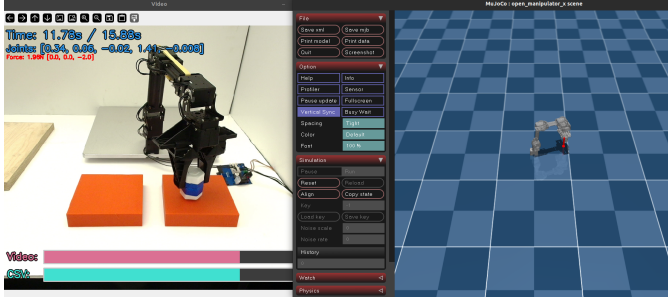
$$\mathcal{L} = w_{\text{pos}}\mathcal{L}_{\text{pos}} + w_{\text{force}}\mathcal{L}_{\text{force}} + w_{\text{gate}}\mathcal{L}_{\text{gate}} + w_{\text{cond}}\mathcal{L}_{\text{cond}} + w_{\text{vel}}\mathcal{L}_{\text{vel}}. \quad (20)$$



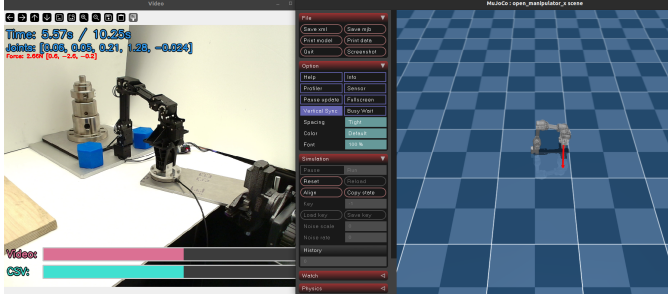
(a) Free motion



(b) Payload-labeled



(c) Mechanically restricted



(d) Force-sensor interaction

Fig. A17: **Representative data-collection configurations.** Free motion, payload supervision, mechanically restricted operation, and fixture-based force sensing are shown from top to bottom.

For the primary OpenManipulator-X configuration, $\mathcal{L}_{\text{pos}} = \mathcal{L}_{\text{arm}} + w_{\text{grip}}\mathcal{L}_{\text{grip}}$ combines Smooth L1 losses on arm joint positions (radians) and the single-finger slide coordinate (millimeters). $\mathcal{L}_{\text{force}}$ is a Huber loss on the predicted base-frame 3D force. It uses $\beta = 0.15$ for the force-sensor run and the final known-weight fine-tuning stage; $\beta = 1$ otherwise. Franka uses a scalar Huber loss with $\beta = 0.15$ on f_z

alone, supervised by the known-payload label $-mg$. Nonzero-force samples receive w_{focal} times the weight of zero-force samples. For the final known-weight fine-tuning stage and the SO-101 runs, the force loss is masked per channel so that only valid labels contribute; other runs supervise all three channels after missing-value sentinels are converted to zero. $\mathcal{L}_{\text{gate}}$ is a binary cross-entropy loss for contact gating. For the OpenManipulator-X motor-condition task, $\mathcal{L}_{\text{cond}}$ is binary cross-entropy on the Joint 3 component of the per-motor output. The condition term is used only for motor-condition tasks, where $w_{\text{cond}} = 100$; otherwise $w_{\text{cond}} = 0$. Selected SO-101 and Franka runs use finite-difference joint velocities for auxiliary supervision, with $w_{\text{vel}} = 5$ on SO-101 and 5 or 10 across the Franka training stages; it is zero otherwise. The OpenManipulator-X default weights are $w_{\text{pos}} = 100$, $w_{\text{grip}} = 0.02$, $w_{\text{force}} = 30$, $w_{\text{gate}} = 1$, and $w_{\text{focal}} = 5$ (3 for the force-sensor run). SO-101 instead uses one equally weighted six-channel pose loss over its five arm joints and rotary jaw. Franka uses seven arm-joint weights $[1, 1, 1, 1, 1.5, 2, 1.5]$ and replays the recorded position-controlled gripper, so the gripper does not contribute to learning the simulator-control surrogate. Each training step uses the features at time t to advance the simulated state toward q_{t+1} ; the directly supervised force, gate, and condition heads are paired with labels at $t+1$. The offline force evaluators instead compare the output computed from the time- t features with the force label time-stamped at t , resulting in a one-sample difference in label alignment between training and evaluation. The condition label is constant within each motor-condition trajectory.

b) Differentiable Simulation: Training uses task-specific truncated rollouts of up to 320 steps. For the primary OpenManipulator-X and SO-101 320-step runs, we begin with 128-step supervision and increase the horizon to 256 and then 320. The no-load runs instead use 128, 192, and 256 steps, motor-condition training remains at 128 steps, and the Franka runs use a fixed 128-step horizon. The simulator uses 4 substeps for OpenManipulator-X, 5 for SO-101, and 8 for Franka. For the torque-actuated OpenManipulator-X and SO-101 simulators, we use direct torque-surrogate prediction; during training, the network output is clipped to platform-specific generalized-effort bounds before it is applied to the simulator: $\tau = \text{clip}(\tau_{\text{net}}; \tau_{\text{min}}, \tau_{\text{max}})$, where $\tau_{\text{net}} = g_{\theta}(\mathbf{X}_t)$. The Franka simulator instead retains the stock affine PD position actuators, and the dynamics head supplies their control channels; its raw output is therefore treated as a simulator-control surrogate rather than a direct torque estimate in N.m. The reported OpenManipulator-X contact and payload rollouts use the same unloaded robot scene with contact dynamics disabled: no measured or predicted external forces, payload bodies, or link-mass changes are applied to the MJX state update. Their generalized effects can therefore be absorbed by the pose-supervised surrogate, whereas the force labels supervise the separate force head directly.

c) Optimization Schedule: The learning rate follows a cosine-annealing schedule from a base rate of 10^{-4} (5×10^{-5} on SO-101) whose decay horizon is set independently of the

epoch budget, so extended runs continue at a small floor rate instead of restarting the decay. The known-weight configuration uses staged warm-starts for the no-gripper and force-refinement stages at 5×10^{-5} and then 5×10^{-6} ; the SO-101 extended configuration is likewise fine-tuned from its scratch run. The three Franka stages use 4×10^{-5} , 1.2×10^{-5} , and 10^{-5} , respectively. The training code maintains both raw and exponential-moving-average parameter tracks.

d) Franka Feature Configuration: The offline Franka force-estimation experiment uses a 52-D input per frame: a seven-joint target pose; current arm positions and velocities; normalized gripper width and goal width; seven commanded torques; motor-side positions and velocities; and seven arm tracking errors plus one gripper error. The recorded commanded-position signal contains only a sparse sequence of stepwise setpoints, so the offline feature construction substitutes the smooth commanded-pose proxy $\tilde{\mathbf{q}}_t^{\text{cmd}} = 1.03 \mathbf{q}_{t+5}^{\text{rec}}$ for the target-position channel. The recorded position-controlled gripper is replayed in simulation. Because this offline proxy is constructed from a future recorded position, the reported Franka force results constitute a future-state-conditioned offline benchmark rather than an online evaluation. In causal deployment, the commanded-pose channel must instead receive the command available at time t from the external controller, teleoperation interface, or inverse-kinematics module.

e) Data and Runtime: For the no-load free-motion benchmark, we train on 80 trajectories (8 per task $\times$ 10 tasks), split across the no-gripper (8-task) and with-gripper (2-task) runs, with a batch size of 16 (12 for the no-gripper fine-tuning stage). Training converges in approximately 1.5 hours on a single NVIDIA RTX 4090 GPU.

C. Visual Supervision through Differentiable Rendering

We use differentiable silhouette rendering to train NeuralActuator together with the hand-eye transform from image-space alignment. Hand-eye calibration estimates the rigid camera-from-base transform $\mathbf{T}_{cb} \in SE(3)$ that projects base-frame robot geometry consistently into the image. Following EasyHeC, the visual objective compares rendered and observed robot silhouettes and backpropagates through rendering, kinematics, differentiable dynamics, and the actuator model. The reported IoU reflects the joint image-space refinement; it does not, by itself, isolate the contribution of the actuator update or establish an improvement in rollout accuracy.

a) Supervision signals: For each selected frame $i \in \mathcal{I}$, the following quantities are available.

Segmentation mask. A binary robot silhouette $M_i \in \{0, 1\}^{H_{\text{img}} \times W_{\text{img}}}$ is obtained using SAM3 [5]. This mask provides pixel-level silhouette supervision in the image plane.

Camera intrinsics. A camera intrinsic matrix $\mathbf{K}_i \in \mathbb{R}^{3 \times 3}$ is estimated using VGGT and analytically transformed back to the original image coordinate system. Camera intrinsics are treated as fixed during optimization.

Camera pose initialization. An initial estimate of the camera-from-base transformation $\mathbf{T}_{cb}^{(0)} \in SE(3)$ is obtained

via a geometric three-point initialization. This initialization aligns depth-lifted image keypoints with corresponding robot keypoints computed from forward kinematics, and serves as the starting point for differentiable refinement.

b) Pose parameterization: We parameterize the camera extrinsic transformation in the Lie algebra. Let

$$\boldsymbol{\xi} \in \mathbb{R}^6, \quad \mathbf{T}_{cb}(\boldsymbol{\xi}) = \exp(\hat{\boldsymbol{\xi}}) \in SE(3), \quad (21)$$

where $\hat{\boldsymbol{\xi}} \in \mathfrak{se}(3)$ is the matrix representation of the twist coordinates and $\exp(\cdot)$ is the matrix exponential. The optimization variable $\boldsymbol{\xi}$ is initialized by applying the logarithm map to $\mathbf{T}_{cb}^{(0)}$.

c) Differentiable actuator rollout and silhouette rendering: For a selected frame i at control step t_i , NeuralActuator predicts the torque surrogate from the input sequence $\mathbf{X}_{t_i}$, and differentiable simulation produces $\mathbf{s}_i^{\text{sim}}(\theta)$. The offline visual experiment uses the synchronized current sequence recorded with the trajectory, so it is recorded-current-conditioned in the same sense as Sec. III. We use the simulated joint configuration and optionally add a small, regularized per-frame correction $\Delta \mathbf{q}_i$:

$$\mathbf{q}_i(\theta) = \mathbf{q}_i^{\text{sim}}(\theta) + \Delta \mathbf{q}_i. \quad (22)$$

Forward kinematics then yields each link pose:

$$\mathbf{T}_{bl,i}^{(l)} = \text{FK}_l(\mathbf{q}_i), \quad l = 1, \dots, N_{\text{link}}. \quad (23)$$

Each link mesh is transformed into the camera frame by $\mathbf{T}_{cb}(\boldsymbol{\xi})\mathbf{T}_{bl,i}^{(l)}$ and rendered with a differentiable rasterizer, producing a soft silhouette $\mathbf{s}_i^{(l)} \in [0, 1]^{H_{\text{img}} \times W_{\text{img}}}$.

Following EasyHeC, individual link silhouettes are aggregated using saturation:

$$\hat{M}_i = \min \left(1, \sum_{l=1}^{N_{\text{link}}} \mathbf{s}_i^{(l)} \right), \quad (24)$$

yielding the rendered soft robot mask $\hat{M}_i \in [0, 1]^{H_{\text{img}} \times W_{\text{img}}}$.

d) Silhouette alignment objective: We minimize the discrepancy between the rendered soft silhouette $\hat{M}_i \in [0, 1]^{H_{\text{img}} \times W_{\text{img}}}$ and the observed binary mask $M_i \in \{0, 1\}^{H_{\text{img}} \times W_{\text{img}}}$:

$$\begin{aligned} \mathcal{L}_{\text{mask}} &= \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \left\| \hat{M}_i - M_i \right\|_F^2 \\ &= \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sum_{u,v} \left(\hat{M}_i(u,v) - M_i(u,v) \right)^2. \end{aligned} \quad (25)$$

e) Joint visual optimization of camera and NeuralActuator: In the reported visual-training experiment, we jointly optimize the NeuralActuator parameters, camera extrinsics, and optional per-frame state corrections. Small per-frame state corrections absorb residual synchronization, sensing, and geometry errors, and are quadratically regularized:

$$\mathcal{L}_{\text{reg}}^q = \lambda_q \cdot \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \left\| \Delta \mathbf{q}_i \right\|_2^2. \quad (26)$$

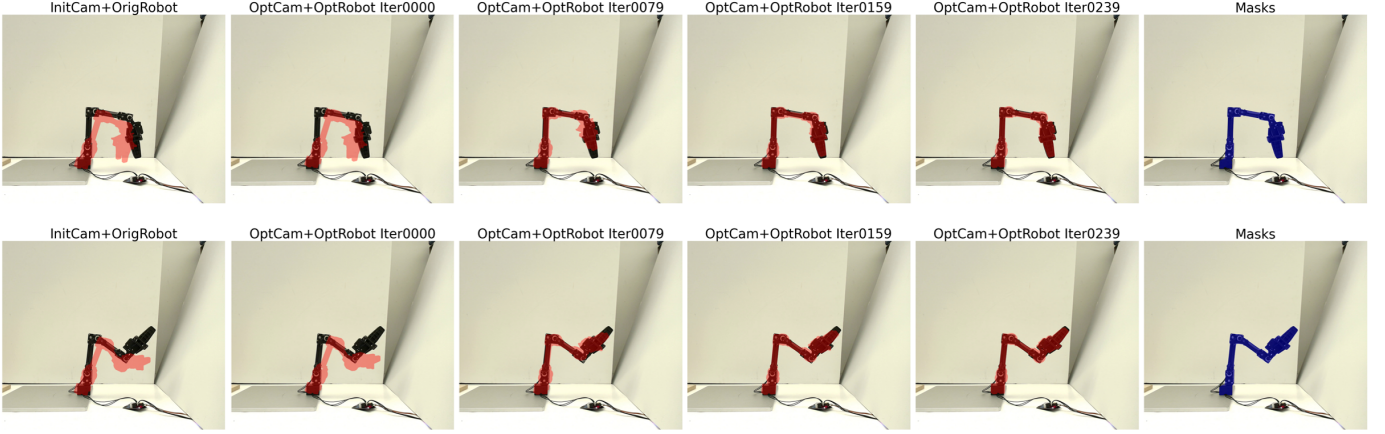


Fig. A18: **Visual refinement through differentiable rendering.** Rows show two robot configurations. Columns show the initial alignment, representative optimization snapshots through iteration 239, and the observed segmentation mask. Red denotes the rendered robot silhouette; blue denotes the observed mask. The complete optimization contains 1,845 gradient steps (Tab. A18).

The visual term is

$$\mathcal{L}_{\text{visual}} = \mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{reg}}^q. \quad (27)$$

Combining the visual term with the trajectory losses in Sec. III, we solve

$$\min_{\theta, \xi, \{\Delta \mathbf{q}_i\}} \mathcal{L}_{\text{traj}} + \lambda_{\text{visual}} \mathcal{L}_{\text{visual}}. \quad (28)$$

We select the positive weights λ_q and λ_{visual} on the training sequence to balance trajectory fidelity, silhouette alignment, and the magnitude of the state corrections, and then hold them fixed for all reported frames. Consequently, gradients from $\mathcal{L}_{\text{mask}}$ propagate to θ through the chain $\hat{M}_i \rightarrow \text{FK} \rightarrow \mathbf{q}_i^{\text{sim}} \rightarrow \text{DiffSim} \rightarrow g_\theta$. Because the camera extrinsics and $\Delta \mathbf{q}_i$ are optimized jointly, the silhouette IoU measures the combined refinement and is not an isolated ablation of the actuator update.

f) Optimization details: We optimize θ , ξ , and the optional state corrections with first-order gradients, use gradient clipping for numerical stability, and retain the lowest-loss solution.

g) Accuracy metrics: We evaluate calibration quality with soft intersection-over-union (IoU) between the rendered soft mask $\hat{M}_i$ and the observed binary mask M_i :

$$\text{IoU}_{\text{soft}} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \frac{\sum_{u,v} M_i(u,v) \hat{M}_i(u,v)}{D_i},$$

$$D_i = \sum_{u,v} \left[M_i(u,v) + \hat{M}_i(u,v) - M_i(u,v) \hat{M}_i(u,v) \right]. \quad (29)$$

h) Quantitative and qualitative results: We report both silhouette alignment accuracy and runtime statistics for joint camera–actuator refinement on a single real-world video sequence. All measurements are conducted on a single NVIDIA A6000 GPU. The evaluation uses six images sampled from a single video; these frames are intentionally non-consecutive

TABLE A17: Silhouette alignment accuracy (IoU) on real-world data.

Method	Mean IoU	Std IoU
Initialization (InitCam + InitRobot)	0.2589	0.0441
Robot refinement (InitCam + OptRobot)	0.2974	0.0754
Joint refinement (OptCam + OptRobot)	0.8515	0.0065

TABLE A18: Runtime statistics per sequence on real-world data.

Method	Iterations	Total time (s)
Joint refinement	1,845	180.23

and correspond to substantially different robot configurations, providing diverse geometric constraints for calibration. In Tab. A17, *OptRobot* denotes the optimized robot-side variables: the NeuralActuator parameters θ and, when enabled, the regularized state corrections $\{\Delta \mathbf{q}_i\}$. The *Robot refinement* row keeps the camera fixed, whereas *Joint refinement* also optimizes the camera extrinsics ξ . Tab. A17 summarizes silhouette alignment accuracy, while Tab. A18 reports the corresponding runtime statistics. Qualitative visualizations of the joint refinement process at different optimization stages are provided in Fig. A18.

D. Network Architecture Comparison

To isolate the effect of architecture while controlling for capacity, we compare models with a budget of approximately 1.4 million parameters. Hidden widths and layer counts are adjusted to keep the MLP, GRU, LSTM, and Transformer variants within 25% of the Transformer parameter count (Tab. A19). All models use the same training data, target definitions, loss weights, augmentation, and evaluation protocol. Unless noted below, they also use a batch size of 16, AdamW with a weight decay of 10^{-4} , and a dropout rate of 0.1. Each input contains eight historical samples and

TABLE A19: **Neural architecture configurations for ablation study.** All models are trained with identical loss weights, data augmentation, and early stopping criteria. Parameter counts are matched within 25% of the Transformer baseline.

Model	Hidden	Latent	Params	LR	Grad Clip
MLP	416	208	1.14M	3×10^{-5}	0.3
GRU	325	162	1.43M	1×10^{-4}	1.0
LSTM	275	137	1.44M	1×10^{-4}	1.0
Ours	192	96	1.44M	1×10^{-4}	1.0

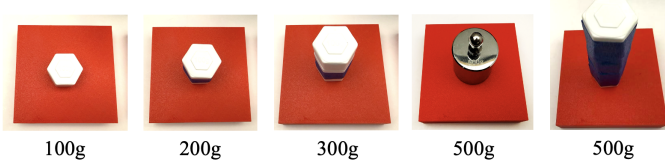


Fig. A19: **Payloads used in the high-level control tasks.**

the current sample, all sampled at approximately 58.8Hz. The MLP uses a lower learning rate (3×10^{-5}) and tighter gradient clipping (0.3) because the larger setting was unstable during differentiable rollouts. NeuralActuator uses four self-attention layers, four heads, hidden dimension 192, feed-forward dimension 384, mean pooling, and query-dependent gated attention. Its encoder feed-forward blocks use GELU, while the output heads use SiLU, matching the implementation used in our experiments.

E. Taxonomy of Robotic Force-Sensing Methods

Tab. A20 summarizes representative approaches for robotic force sensing and contact inference, grouped by sensing signals and whether they explicitly estimate force magnitude. Classic sensorless collision monitoring infers external torques from residuals/momentum observers and mainly supports contact detection [10, 11, 17]. Recent intrinsic and whole-body touch methods exploit joint-torque and position sensing, whereas learning-based force-aware policies span low-cost bilateral-control arms and force-sensor-free whole-body robots [26, 14, 29, 52]. For manipulators, virtual force sensing, contact localization, and observer/filter-based methods can estimate interaction forces or localize contacts, generally using torque feedback, commanded torque, or calibrated dynamics [37, 38, 23, 33, 18, 31, 39]. Visual cues can estimate contact-force location and magnitude, whereas inertial-aided proprioceptive methods improve collision monitoring [48, 2]. In contrast, NeuralActuator enables force-magnitude estimation on low-cost servos by learning a simulator-equivalent torque surrogate from actuation telemetry and jointly predicting external forces without dedicated torque sensors.

F. Imitation Learning

We compare two approaches for training the high-level controller via behavior cloning. For the **position-only baseline**, the per-step observation is $\mathbf{o}_t^{\text{pos}} = [\mathbf{q}_{1:4,t}, a_t]^\top \in \mathbb{R}^5$. The policy receives the eight-step history $\mathbf{O}_t^{\text{pos}} = (\mathbf{o}_{t-7}^{\text{pos}}, \dots, \mathbf{o}_t^{\text{pos}}) \in \mathbb{R}^{40}$ and minimizes $\mathcal{L}_{\text{BC}} = \|\pi_\psi(\mathbf{O}_t^{\text{pos}}) - \mathbf{g}_t^*\|_2^2$, where $\mathbf{g}_t^* \in \mathbb{R}^5$ is the demonstrated command for the four arm

joints and gripper aperture. Our force-aware controller augments the input with the estimated end-effector external force $\hat{\mathbf{f}}_t$ predicted by a *pretrained* NeuralActuator, giving $\mathbf{o}_t^{\text{force}} = [\mathbf{q}_{1:4,t}, a_t, \hat{\mathbf{f}}_t]^\top \in \mathbb{R}^8$ and the corresponding eight-step history $\mathbf{O}_t^{\text{force}} \in \mathbb{R}^{64}$, which replaces $\mathbf{O}_t^{\text{pos}}$ in the same BC objective. NeuralActuator is trained once on the full payload set $\{100, 200, 300, 400, 500\}$ g and is kept *frozen* during policy learning. The BC-policy demonstrations also include the 500 g condition used in the pick-and-place evaluation. The downstream protocol therefore evaluates task-level transfer within the payload range represented during pretraining. For offline force analysis, we compute a trajectory-level statistic on each demonstration-aligned rollout:

$$\mathcal{R}_f = \frac{1}{T_r} \sum_{k=0}^{T_r-1} \|\hat{\mathbf{f}}_{t+k}\|_2^2. \quad (30)$$

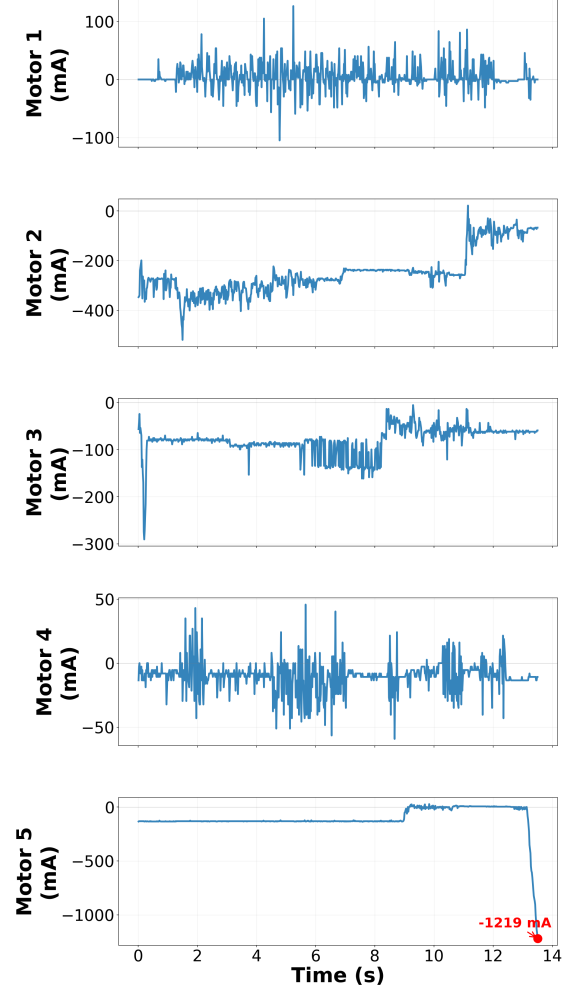


Fig. A20: **Current overload in position-only behavior cloning.** Representative motor-current traces during high-level execution. The position-only baseline exhibits over-current spikes beyond the 1200 mA safety limit, which triggers the onboard protection and terminates the rollout.

Here, T_r is the rollout length. These offline rollouts re-

play the synchronized current measured in the corresponding demonstration at every step. Because NeuralActuator does not predict the current that would result from a counterfactual policy action, $\mathcal{R}_f$ is a diagnostic rather than an additional policy-training loss; the policy is optimized with $\mathcal{L}_{BC}$. During hardware execution, $\hat{\mathbf{f}}_t$ is recomputed causally from telemetry available through time t , including the live current $\mathbf{i}_t$. Both methods use identical Transformer policies (4 layers, hidden dimension 128, and 4 heads) and are trained for 5,000 epochs on the same demonstrations. For policy training, we use payloads $\{100, 200, 300, 500\}$ g with gripper contact points at approximately matched heights, and evaluate on 400 g and 500 g loads. The position-only baseline can produce current spikes above the 1200 mA safety threshold, triggering onboard protection and terminating the execution early (Fig. A20).

Direction	Works	Signals	Force mag.	Typical regime
Residual/momentum collision monitoring	[10, 11, 17]	$q, \dot{q}, \tau_c$ / TS	Partial	Simulation / torque-sensed robots
Intrinsic/whole-body touch from proprioception	[26, 14]	TS + q	Partial	High-end (torque-capable)
Learning-based force-aware policies	[29, 52]	ILBiT: $q, \dot{q}, \hat{\tau}$; Zhi: state history	Partial / Yes	Low-cost arm / legged robots
Arm contact localization / virtual force sensing	[37, 38]	Magrini: RGB-D + $q, \dot{q}, \tau$; Manuelli: $q, \dot{q}, \tau$	Yes / Partial	Torque-capable / calibrated dynamics
Observer/filter for sensorless force estimation	[23, 33, 18, 31, 39]	$q, \dot{q}$ + TS / $I \rightarrow \tau$	Yes	Industrial / Mixed
Vision- and inertial-aided force/contact inference	Vision: [48] IMU: [2]	Wang: Vision + $q, \dot{q}$; Birjandi: IMU + q	Yes / Partial	Mixed (observability-dependent)
NeuralActuator (ours)	This paper	Telemetry + $q, \dot{q}$	Yes	Low-cost (servo-driven)

TABLE A20: **Taxonomy of robotic force sensing and contact inference.** **Signals:** TS = torque sensing; $I \rightarrow \tau$ = torque inferred from current/effort; $\hat{\tau}$ = simulator-equivalent torque surrogate; $q, \dot{q}$ = kinematics; τ_c = commanded torque; $\hat{\tau}$ = estimated torque; Vision = exteroception; IMU = inertial sensing. **Force mag.:** *Yes* = explicit force/wrench magnitude; *Partial* = mainly detection/localization or residual signals. NeuralActuator predicts $\hat{\tau}$ from actuation telemetry and estimates external forces on low-cost servos without dedicated torque sensors; the torque-surrogate and force heads are parallel outputs, and $\hat{\tau}$ is not an input to the force head.

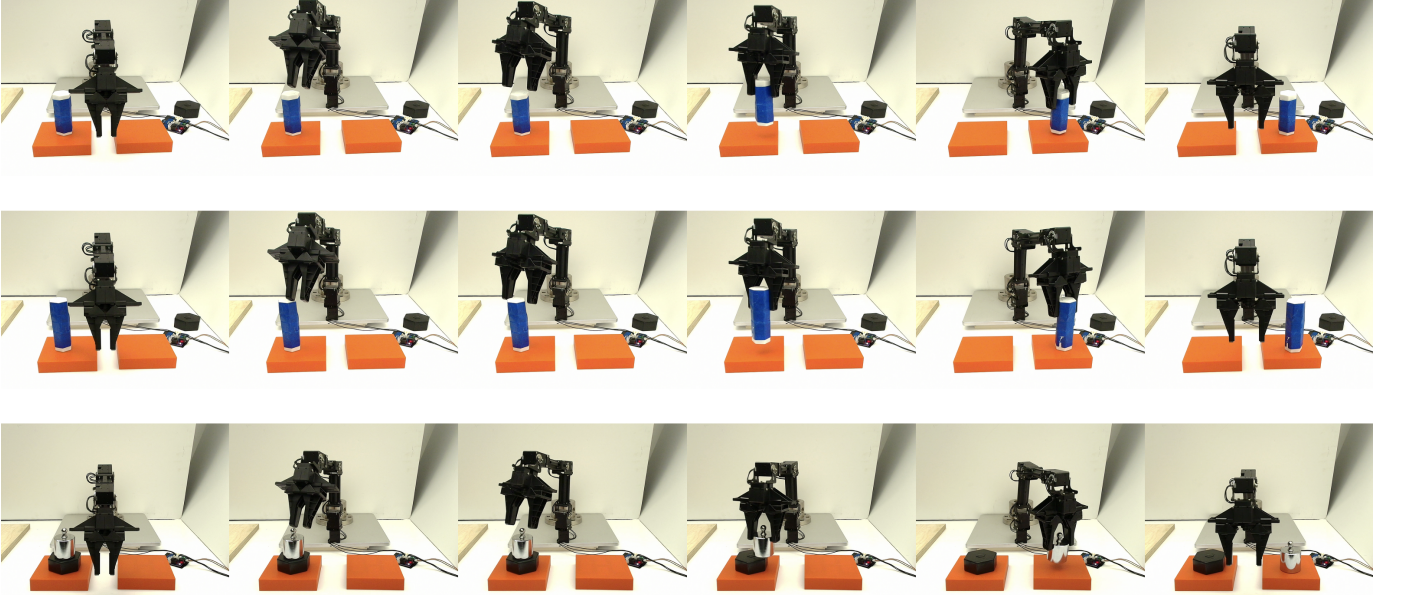


Fig. A21: **Qualitative results on high-level pick-and-place.** Time-lapse snapshots (left to right) of the robot grasping, lifting, transporting, and placing objects under different payloads. From top to bottom, we evaluate 400 g, 500 g, and a differently shaped 500 g weight.