$\mathcal{N}_0$-VTLA: Scaling Vision–Tactile–Language–Action Model with Latent Tactile Tokens

NeoteAI Team & Fudan TEAI Team

We present $\mathcal{N}_0$-VTLA, a vision–tactile–language–action (VTLA) foundation model capable of (1) fine-grained contact-rich manipulation with tactile perception and tactile-feedback control, and (2) offline policy improvement from stored deployment data. Stepping towards current visual-based backbones, we propose an overall training recipe for tactile integration, consisting of visuo-tactile pre-training, staged tactile-pathway integration, and advantage-conditioned offline policy improvement. During pre-training, the policy learns broad contact priors from NeoData, our large-scale visuo-tactile robot dataset. To our knowledge, $\mathcal{N}_0$-VTLA is the first VTLA model pretrained on tactile data at scale. During post-training, we augment the policy with a predictive tactile pathway, distilling the contact patterns learned at scale into the fine motion adjustments in downstream tactile-centric manipulation. For offline policy improvement, we introduce ALTER, an advantage-conditioned offline Reinforcement Learning (RL) method that converts relative progress and trajectory-event comparisons into binary advantage labels for policy training on a fixed deployment corpus. This procedure further improves task-specific learning on contact-rich skills such as deformable object manipulation. Across contact-rich benchmarks, $\mathcal{N}_0$-VTLA outperforms strong baselines by wide margins: it wins all nine real-robot NeoReal tasks and reaches 63.8% mean success on the twenty-task simulation suite against 44.0% for the strongest baseline. $\mathcal{N}_0$-VTLA policies trained with ALTER reach 75–95% success on three long-horizon real-robot tasks. Results lay a foundation for versatile tactile-driven manipulation policies.

Date: July 26, 2026
Code: https://github.com/neoteai/N0-VTLA
Website: https://research.neoteai.com/n0-vtla/

NeoteAI
Institute of Trustworthy Embodied AI, Fudan University

1 Introduction

Vision–language–action (VLA) models have made manipulation policies general. Fine-tuned from pretrained vision–language backbones, they follow instructions and transfer across tasks, scenes, and embodiments [9, 37, 58]. Touch, however, has remained largely absent from this progress, leaving current policies with a persistent weakness in contact-rich manipulation: the tactile extensions attempted so far train on task-scale collections, at most tens of hours gathered for a handful of skills. This report presents, to our knowledge, the first VLA policy pretrained on tactile data at scale. We scope the report to vision-based tactile sensing [39, 86], instrumenting each gripper finger with our self-developed sensor so that contact is read out as an image. This signal is nothing like a camera view: tactile frames are noisy, nearly empty away from contact, and informative almost only in the brief windows when contact forms or is about to change.

Existing systems integrate touch along one of two paths. The first concatenates tactile tokens into the vision–language context and treats the tactile stream as one more camera [22, 36]; yet a signal that is sparse and mostly silent buys little in a prefix built for information-dense views. The second injects the current tactile reading into the action pathway to guide denoising [77, 91]; this placement mischaracterizes the role of touch, since a tactile frame records contact that actions already taken have produced and, by itself, says little about the contact the next actions must anticipate. Conditioned on it alone, the policy stays one step behind its own contact events.

$\mathcal{N}_0$-VTLA, which we read as NeoVTLA, takes a third path: it keeps tactile out of the vision–language prefix and conditions the action expert on a prediction of touch rather than the current reading. A small predictor reads the vision–language context together with the current tactile tokens and emits *latent tactile*

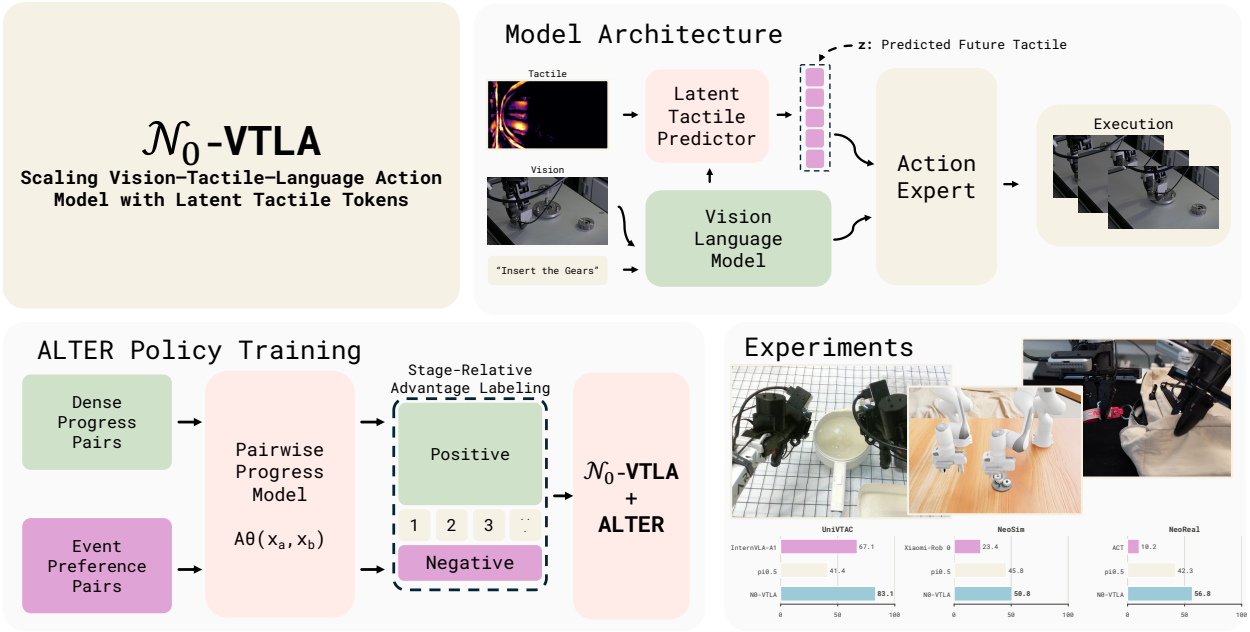

Figure 1 **$\mathcal{N}_0$-VTLA at a glance.** $\mathcal{N}_0$-VTLA encodes vision, the instruction, and tactile difference images, predicts latent tactile tokens z, and conditions a flow-matching action expert on them. Beyond demonstrations, ALTER converts deployment experience into stage-relative advantage labels for offline policy learning. The experiment panels plot the headline means against the base VLA policy and the strongest specialist baseline on UniVTAC [18], NeoSim, and NeoReal. $\mathcal{N}_0$-VTLA leads every panel.

tokens z that estimate the net tactile change over the coming action chunk, so the policy acts on the contact its own actions are about to cause. The tactile frames are contact-difference images encoded by a frozen pretrained visual encoder through a lightweight trainable projection, and a three-stage recipe brings this newly initialized pathway online, as Figure 1 shows. Beyond supervised task adaptation, we formulate learning from stored deployment experience as advantage-conditioned offline RL. ALTER trains a pairwise progress model from clean demonstrations, tactile-detected object-drop events, and logged human corrections, then assigns stage-relative binary advantage conditions for policy learning.

The full system rests on a vision–language–action backbone built on PaliGemma [6], a canonical cross-embodiment action space, and NeoData, a multi-platform visuo-tactile corpus spanning single and dual-arm configurations, documented in a companion data report [51]. Before evaluating the full system, we verify the latent pathway itself: after Stage 1, the latent tokens retrieve their matching future-tactile targets at 92.3% top-1 accuracy, where chance sits at 3.2%. Whether this grounded representation translates into better task performance is the more demanding test. We evaluate $\mathcal{N}_0$-VTLA against external baselines on the NeoReal real-robot benchmark and the simulated contact-rich suite under identical protocols. On the twenty-task simulation suite $\mathcal{N}_0$-VTLA leads the strongest baseline by a wide margin, and it wins all nine real-robot NeoReal tasks.

In summary, this report makes three contributions:

- **Large-scale tactile pretraining (§2.1).** $\mathcal{N}_0$-VTLA is pretrained on NeoData, the large-scale visuo-tactile robot data across multiple robot platforms, made trainable as one model by a canonical cross-embodiment action space and a quality-verified data pipeline [51].
- **Latent tactile tokens (§2.2).** Touch is treated as a prediction target rather than as observation context, a predictor estimating the tactile change over the coming action chunk and conditioning the action expert directly, brought online stably by a three-stage recipe.
- **Offline policy improvement with ALTER (§4.4).** A pairwise progress model, supervised by tactile-grounded stage annotations, tactile-detected object-drop events, and logged human corrections, produces stage-

relative advantage labels for offline policy learning. The method applies to both the base VLA policy and $\mathcal{N}_0$-VTLA, with $\mathcal{N}_0$-VTLA+ALTER achieving the highest success on all three tasks.

2 Model

$\mathcal{N}_0$-VTLA is a policy for contact-rich manipulation. It reads camera views, a language instruction, robot state, and touch, and it generates a chunk of future actions. The model consists of a pretrained vision–language–action backbone built on PaliGemma [6] and one added component, a latent tactile pathway between perception and action. The backbone carries the views, instruction, and state in its vision–language prefix and generates the action chunk with a flow-matching action expert. In the added pathway, a frozen-backbone tactile encoder turns each finger's contact-difference image into tokens, and a small predictor distills those tokens, in the context of the scene and instruction, into *latent tactile tokens* z that estimate the net contact change expected over the coming action chunk. The action expert is conditioned on z directly, and tactile never enters the vision–language prefix. Touch therefore enters the policy as a prediction target rather than as one more observation. Figures 2–4 lay out this design as a three-step recipe, and the section follows them. Section 2.1 fixes the base policy, and Section 2.2 walks through the tactile pathway step by step.

2.1 Base Architecture

The base policy pairs a PaliGemma vision–language backbone [6] with a flow-matching action expert [43]. Camera views, the instruction, and the robot state form the model *prefix*, state entering that prefix in discretized form rather than as a separate continuous input. Conditioned on the prefix, the expert denoises an action chunk over a horizon of $H = 50$ steps in the canonical 32-dimensional container of Section 3, whose width and slot layout we inherit unchanged from the pretrained action head so that its weights load directly. The flow-matching objective is unmasked over all 32 dimensions. Each platform populates the dimensions its embodiment uses, the rest carry zero targets the model learns to reproduce, and single- and dual-arm data therefore coexist in one model under a single fixed-width objective. All other aspects, including architecture, tokenization, and training procedure, are inherited unchanged from the pretrained backbone.

2.2 Latent Tactile Tokens

Figure 2 shows Step 1, in which the tactile predictor is trained against a future-tactile target. Figures 3 and 4 show Steps 2 and 3, in which the action expert first learns to consume the resulting latents while the vision–language pathway is masked, and the full policy then trains end to end. The model that leaves Step 3 is the deployed controller. The paragraphs below introduce each component in the same order.

The tactile encoder. Every panel of both figures begins the same way. The policy never sees a raw tactile frame. For view k we subtract the episode-start baseline frame tac_0^k, the zero-contact reference established in Section 3, from the current frame tac_τ^k in pixel space, and encode the difference with a frozen self-supervised visual encoder, followed by a trainable linear projection to the shared token width d of the vision–language backbone:

$$g_k = f_{\mathrm{enc}}\big(\mathrm{tac}_\tau^k - \mathrm{tac}_0^k\big) \in \mathbb{R}^{10\times d}, \tag{1}$$

where f_{enc} denotes the frozen encoder composed with the trainable projection. Each tactile image yields 10 tokens, one class token and nine spatial tokens from a 3×3 adaptive average pool over the encoder's 16×16 patch grid, and the tokens of the n active views are concatenated into $g = [\, g_1; \ldots; g_n \,] \in \mathbb{R}^{10n\times d}$. Differencing against a per-episode baseline, rather than encoding the absolute gel image, removes the static gel appearance and much of the mount-specific imprint, making the representation robust to, though not strictly invariant under, differences in sensor placement. Freezing the encoder is deliberate. It preserves the self-supervised representation intact, it lets a previously unseen sensor be onboarded by training only the lightweight projection, and it removes the encoder's activations and optimizer state from the training memory budget.

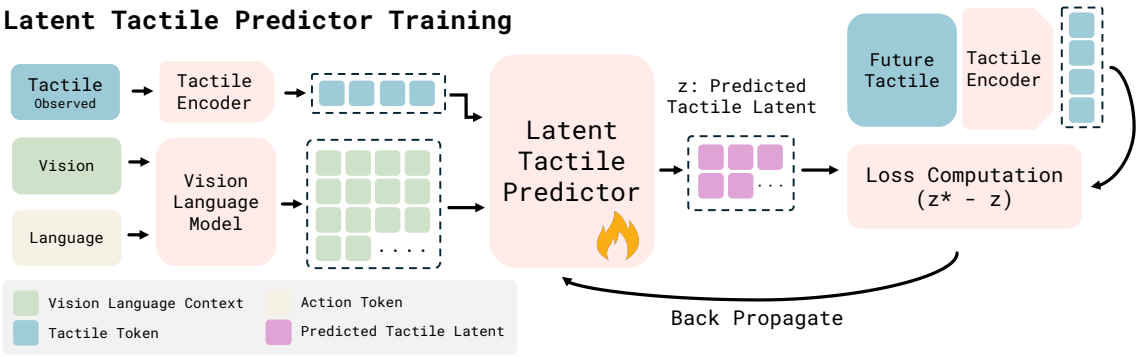

Figure 2 Step 1: latent tactile predictor training. The current tactile difference is encoded into tokens g. The predictor reads g together with the contextualized vision–language prefix and emits the latent tactile tokens z. The future-tactile target z^* comes from the same tactile encoder applied to the coming tactile change, and the loss on z^* and z backpropagates into the predictor alone.

Step 1: the predictor and its future-tactile target. The predictor is a lightweight module that reads the current tactile difference tokens g in the context of the scene and instruction, carried by the contextualized vision–language prefix, and distills them into a compact set of learned latent queries that become the latent tactile tokens z. When an episode carries no tactile at all, a learned null token stands in for g, so z is always produced and the policy falls back to vision–language control rather than failing. What makes z predictive rather than merely descriptive is the target Figure 2 attaches to it,

$$z^* = \frac{1}{n}\sum_{k=1}^{n} f_{\text{enc}}\big(\text{tac}_{\tau+H}^{k} - \text{tac}_{\tau}^{k}\big) \in \mathbb{R}^{10\times d}, \qquad H = 50, \tag{2}$$

obtained by applying the same tactile encoder to each view's tactile change over the next H steps and averaging the resulting tokens across the n active views. The predictor output $z \in \mathbb{R}^{10\times d}$ is trained to match z^*. The supervision combines a symmetric InfoNCE[66] contrastive loss that pulls the predicted latent toward its matching future-tactile target and an auxiliary L_1 reconstruction of a coarse future-tactile-difference field. This is the supervision with which the three-stage recipe grounds the predictor. A *free-latent* simplified variant omits the Step 1 supervision entirely, shaping z through the action gradient alone. It requires no future frames during training.

Step 2: conditioning the action expert. The latent tokens z are projected to the action-expert width and prepended to the action suffix, ahead of the noisy action tokens. The current-contact tokens g never enter the action expert. They reach action generation only through the predictor, which distills them into z. The latent tokens form their own conditioning block. The action tokens attend to z and, as in the base policy, to the vision–language prefix, the prefix never attends back to the latent tokens, and the z positions are sliced off before the action output head so that they never emit actions. Because the pretrained expert has never consumed such a token, Step 2 (Figure 3) trains this interface in isolation. The vision–language pathway is masked so that action prediction must draw on z, aligning the latents with the expert before anything else moves.

Step 3: training the full policy end to end. Step 3 (Figure 4) removes that mask and opens the whole policy to joint training. The direct prefix-to-expert path is restored, so the expert again sees scene and instruction alongside z. Every component except the frozen tactile encoder backbone then adapts under the action objective. What is frozen at each step, and why the order matters, is the subject of the training chapter.

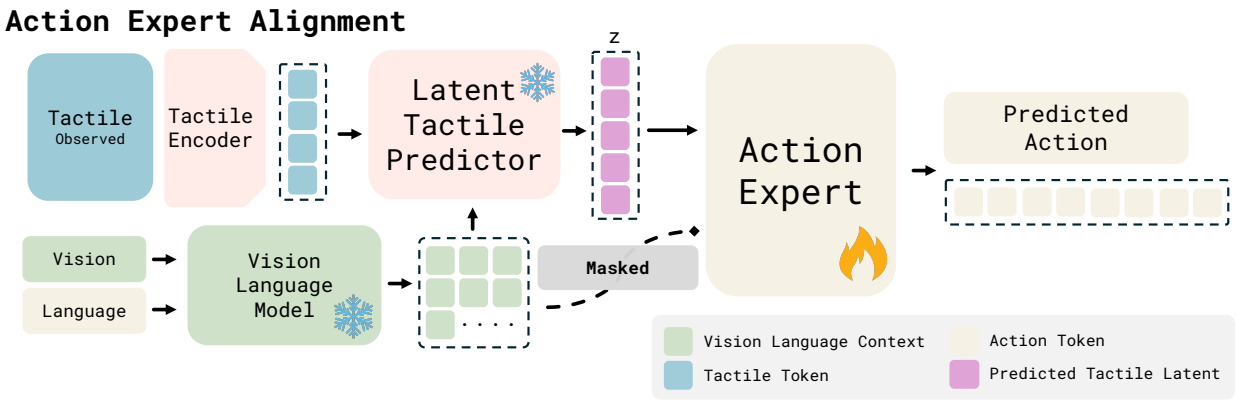

Figure 3 Step 2: aligning the action expert with the latent tokens. With the predictor and the vision–language backbone frozen, the vision–language context is masked before the action expert, so action prediction must draw on the latent tactile tokens z while the expert learns the interface.

Why predict, not react? The design choice is what conditions the action expert. Handing the policy its current tactile reading, whether concatenated into the prefix or injected alongside the actions, hands it a record of contact already made, and spends capacity built for information-dense views on a signal that is sparse and mostly silent. The latent predictor instead asks the model to form an explicit internal estimate of the contact state, and, under supervision, of the net contact change expected over the chunk horizon. It then conditions the action head on that estimate. The behaviors where tactile matters most are anticipatory, such as the millimeter-scale pre-load before a grasp closes and the catch of incipient slip before the object moves. These live in the pre-contact blind spot, where a purely reactive signal arrives too late to shape the action that caused it. We therefore hypothesize that conditioning actions on a predictive latent, rather than on raw current tokens, is the better inductive bias for contact-rich control. This choice to predict in a learned latent space rather than reconstruct raw sensory signals follows the joint-embedding predictive principle [1] explored for self-supervised representation learning in LeJEPA [4] and for latent world modeling from pixels in LeWorldModel [49].

Two observations ground this design choice. What decides a 50-step chunk is the contact the chunk itself is about to create, and no encoding of the present frame contains it. Ranking future-tactile targets by the current tactile encoding alone retrieves 57% top-1 where the predictor reaches 92.3%, with the margin widening as the candidate pool grows, as Section 5.5 details. And the prediction objective pins the latent to touch before the policy ever optimizes through it, where features shaped by the action gradient alone would be free to drift into an appearance cue rather than contact state.

3 Data

$\mathcal{N}_0$-VTLA is pretrained on NeoData [51], our large-scale curated multi-platform visuo-tactile corpus, spanning single- and dual-arm robot manipulators as well as a UMI-style handheld collection gripper [24]. Every gripper finger that participates in a manipulation task carries our self-developed visuo-tactile sensor, so contact is read out as a stream of tactile images rather than as a low-dimensional force signal. Collection protocols, corpus composition, and sensor specifications are documented in the companion data report [51]. This section states only the conventions the rest of the report depends on.

Canonical action and state schema. All embodiments are unified into one fixed 32-dimensional state and action container, inherited from $\pi_{0.5}$ [58]. The container is laid out for two arms, with the first 20 dimensions split into one 10-dimensional slot per arm and the remaining 12 left unused and always zero. Within a slot, the 10 dimensions comprise a 3-dimensional end-effector position, a 6-dimensional rot6d

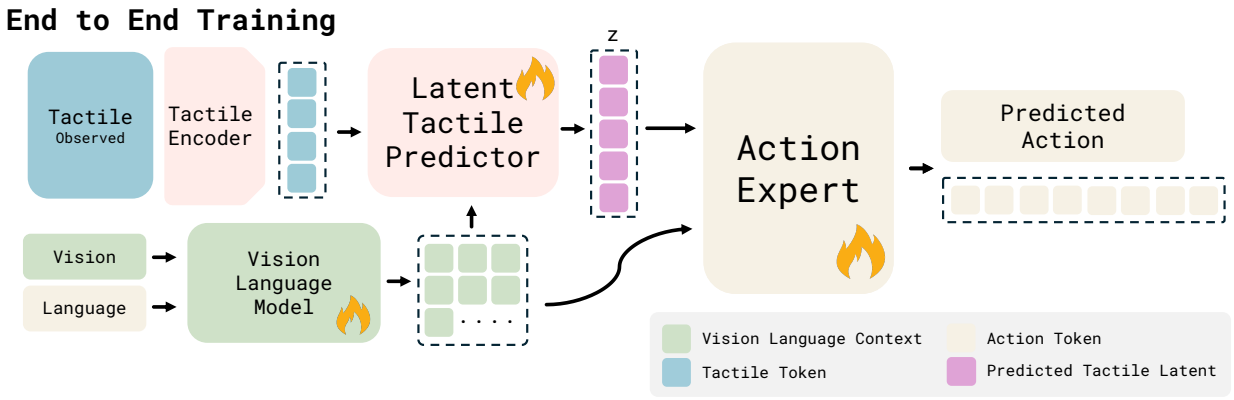

Figure 4 Step 3: end-to-end training. Everything except the tactile encoder backbone unfreezes and the full policy trains end to end, emitting the predicted action chunk. The free-latent simplified variant omits the Step 1 supervision, shaping z through the action gradient alone.

rotation [95], and a 1-dimensional gripper channel. Dual-arm episodes populate both slots. Single-arm episodes populate the first slot only and leave the second zero-filled as well. Single- and dual-arm data therefore coexist in one fixed-width container, and what the policy does with the zero-filled dimensions is fixed by the objective of Section 2.1. Actions are stored as absolute end-effector poses. At training time each chunk is rewritten relative to its own first frame, so the model predicts motion relative to the pose at which the chunk begins. At deployment the predicted chunk is mapped back to absolute poses through the inverse of that transform, and inverse kinematics resolves those poses into the joint commands sent to the robot. Normalization statistics are computed on the chunk-relative representation, separately for each pairing of robot and action schema.

Tactile collection convention. Each participating gripper finger contributes one tactile stream, captured on the same clock as the RGB and proprioceptive channels. The per-platform stream counts and the common frame rate are listed in the data card of Appendix A. Each episode begins with a short zero-contact baseline, the gripper open and static for at least 0.5 s. That baseline frame is the episode's zero-contact reference, and every tactile frame recorded afterwards is interpreted relative to it.

Data quality verification. Every converted repository is verified for data quality before it enters training. Verification checks that a repository is complete and internally valid, that its stored conventions match the schema above, and that its statistics and media are consistent with what the training pipeline assumes. Repositories that fail are repaired or excluded rather than trained on. The individual invariants, and the symptom each produces when it is violated, are catalogued in Appendix C.

Simulated data. Simulated data enters through the same door. Episodes from the UniVTAC visuo-tactile simulator [18], which supplies the NeoSim suite evaluated in Section 5.3, are converted into the canonical schema above and verified alongside real data, so that one policy interface applies to both.

4 Training

$\mathcal{N}_0$-VTLA reaches deployment through three core phases. A three-stage recipe then brings the latent tactile pathway online. Supervised post-training specializes the resulting generalist to individual tasks. After this core recipe, an optional procedure post-trains the task policy on its own deployment data. Throughout, the trainable surface grows only after each new interface has been grounded, so the tactile pathway comes online without destabilizing the pretrained policy.

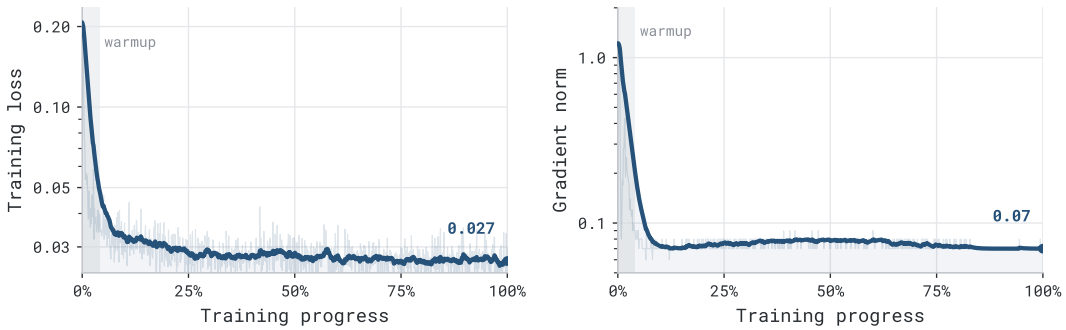

Figure 5 Multi-platform visuo-tactile pretraining. Training loss and gradient norm over multi-platform visuo-tactile pretraining. The loss descends smoothly and gradient norms stay flat throughout, consistent with the pretrained initialization transferring cleanly to the visuo-tactile action space.

4.1 Base Pre-training

Base pre-training is conducted at cluster scale on the NeoData corpus. Stability at this scale is achieved by design, through choices validated in controlled comparisons, and is visible in the smooth loss and flat gradient norms of Figure 5.

4.2 Three-Stage Latent-Tactile Training

The tactile pathway, newly initialized, attaches to the pretrained multi-platform checkpoint, and the three steps of Figures 2–4 bring it online, each stage proceeding from the checkpoint the previous one produces. The free-latent configuration corresponds to collapsing this recipe into a single joint stage with no auxiliary supervision.

Stage 1: grounding the predictor. With the entire base policy frozen, we train only the predictor, the tactile projection, and a lightweight reconstruction head. The predictor output z is pulled toward the future-tactile target z^* of Eq. 2 by a symmetric InfoNCE[66] objective. Write $h(\cdot)$ for the mean pooling over the ten latent tokens followed by ℓ_2 normalization, and, for a batch of B samples,

$$s_{ij} = \langle h(z_i),\, h(z_j^*) \rangle \tag{3}$$

for the cosine similarity between the i-th prediction and the j-th target. The contrastive term

$$\mathcal{L}_{\mathrm{NCE}} = -\frac{1}{2B} \sum_{i=1}^{B} \left[\log \frac{e^{s_{ii}}}{\sum_{j=1}^{B} e^{s_{ij}}} + \log \frac{e^{s_{ii}}}{\sum_{j=1}^{B} e^{s_{ji}}} \right] \tag{4}$$

matches each predicted latent to its own future target, taking the other targets in the batch as negatives, and is symmetrized over both retrieval directions. In parallel a reconstruction head r_ψ decodes z back to a coarse future-tactile-difference field and is trained with an ℓ_1 term $\mathcal{L}_{\mathrm{rec}} = \| r_\psi(z) - \bar{D}_{\tau \to \tau+H} \|_1$ against the same horizon, where $\bar{D}_{\tau \to \tau+H}$ is the downsampled contact-change field over the coming chunk. The stage minimizes

$$\mathcal{L}_1 = \mathcal{L}_{\mathrm{NCE}} + \lambda_{\mathrm{rec}}\, \mathcal{L}_{\mathrm{rec}}, \tag{5}$$

where $\lambda_{\mathrm{rec}} > 0$ balances the two terms. The contrastive term supplies the discriminative pressure that makes z retrieve the right future contact; the reconstruction term anchors it to the spatial layout of that contact, discouraging a shortcut latent that separates batches without encoding where contact forms. Because gradients touch only the shallow predictor stack, the stage is cheap and converges quickly. At convergence the latent tokens are strongly grounded in touch: the latent z retrieves its matching future-tactile target with 92.3% top-1 accuracy against a 3.2% random baseline, analyzed in Section 5.

Stage 2: aligning latents with the action expert. The pretrained action expert has never consumed a latent tactile token, so we next teach it the interface. We hold the tactile perception stack frozen at its Stage 1 checkpoint and train only the latent-to-expert projection and the action expert, under the base action objective. Concretely, in the expert's attention the keys and values from the vision–language prefix are masked out for the action queries, leaving the latent tokens z and the noised action tokens as the only conditioning the expert can attend to. Masking the prefix removes the shortcut of predicting actions from scene and instruction alone, so the only route to lowering the action loss runs through z. The expert learns to read touch through z before any joint training loosens the rest of the policy, in the spirit of the staged alignment strategies explored for language–action models [67].

Stage 3: end-to-end joint training. With the predictor grounded and its interface aligned, we unfreeze everything except the always-frozen tactile encoder backbone and train the full policy jointly under the standard pre-training recipe, on the action objective alone. The vision–language mask of Stage 2 is removed, so the direct prefix-to-expert path is restored and the expert again sees scene and instruction alongside z. Gradients from the action objective now flow together through the predictor, the two projections, the action expert, and the vision–language backbone, letting the perception stack adapt to what the expert actually needs. The contrastive and reconstruction targets of Stage 1 are no longer applied. The predictor keeps its grounding through the action gradient alone, which the perturbation probe of Section 5.5 confirms it retains rather than reroutes.

Why three stages? The staging turns one hard joint optimization into a curriculum, each stage handing the next a better starting point. Stage 1 fixes what z *means*: trained only against the future-tactile target, the bottleneck is forced to encode contact rather than a second copy of the prefix it already sees. Stage 2 fixes how the expert *uses* z: with the prefix masked, the only way to lower the action loss is to read touch through z, so the expert commits to the new interface while the rest of the policy stays put. Stage 3 then trains everything jointly, but from an initialization where z is already grounded and already consumed, so joint optimization refines a working pathway instead of having to discover grounding and interface at once. The free-latent variant collapses all three into a single joint stage: with no Stage 1 target, the action gradient alone shapes z, and nothing stops the bottleneck from degenerating into a copy of the prefix or being bypassed altogether. After joint training the latent stays tactile-leaning (Section 5.5), consistent with Stage 3 refining the grounded pathway rather than rerouting around it. A randomly initialized tactile pathway can therefore be grafted onto a pretrained VLA and brought online without destabilizing it.

4.3 Supervised Task Adaptation

We adapt a pretrained $\mathcal{N}_0$-VTLA checkpoint to each downstream task by supervised fine-tuning(SFT) on a few hundred demonstrations, warm-starting from that checkpoint and reusing the pre-training recipe at reduced scale. Normalization statistics are always recomputed on the task's own data and never reused from pre-training, per Section 3. The same recipe covers both real-robot tasks and the simulated task suite, with one policy per task.

4.4 Offline RL from Deployment Data with ALTER

We call our method ALTER, short for Advantage Labeling from Trajectory Events and Relative Progress. Given a fixed deployment corpus, ALTER performs advantage-conditioned offline RL [57] without additional environment interaction, as summarized in Figure 6. Clean demonstrations provide dense progress supervision from signal-grounded stage intervals, whose boundaries are localized using tactile contact changes, end-effector kinematics, gripper state, and visual event cues. Imperfect deployment trajectories instead provide sparse before-and-after preferences from tactile-detected object-drop events and logged human-in-the-loop corrections. These dense and sparse signals jointly train a task-specific pairwise progress model. We then freeze the model and apply it to every trajectory retained for offline policy learning. Comparing each observation with the episode start estimates global task phase, while comparing it with the observation

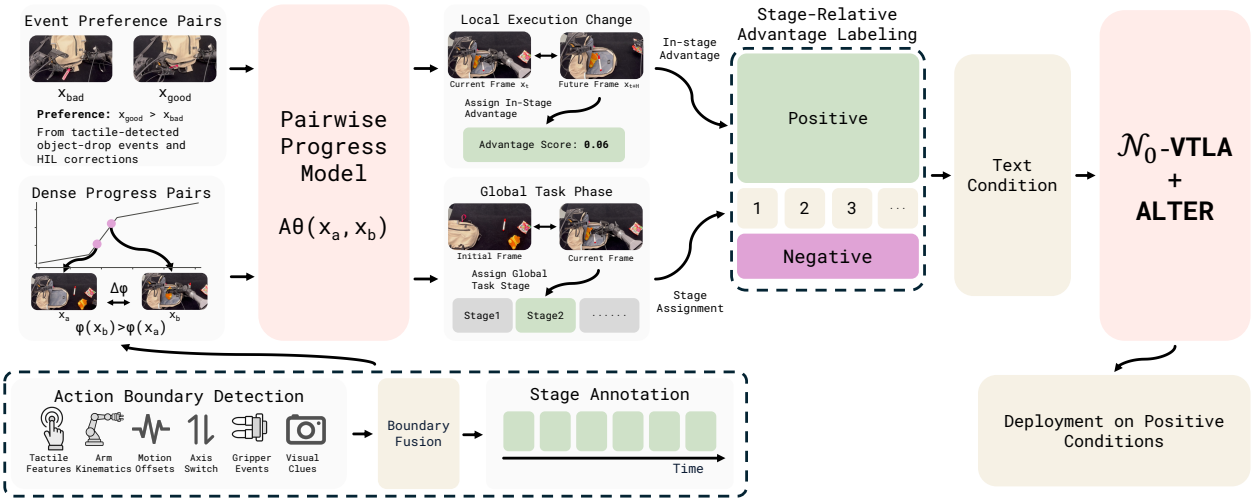

Figure 6 ALTER for offline policy improvement. Tactile contact changes, complemented by kinematic and visual cues, ground stage annotations of clean demonstrations and yield dense progress pairs. Tactile-detected object-drop events and logged HIL corrections yield sparse preference pairs. Both supervise a pairwise progress model, which is then frozen to estimate global task phase and local execution change for each stored trajectory. Within each predicted stage, local-change estimates produce binary advantage labels that are appended to the task prompt during policy training. At deployment, the task prompt uses the positive label.

one action-chunk later estimates local execution change. Within each predicted stage, we rank samples by estimated local change, assign positive labels to higher-ranked samples and negative labels to lower-ranked ones, and append the resulting label to the task prompt used for policy training.

Deployment corpus and offline annotations. After deploying task-adapted policies, we retain clean teleoperation demonstrations, autonomous rollouts, and human-in-the-loop (HIL) rollouts. The HIL corpus also includes staged-recovery episodes that start from selected error states and record human-teleoperated recovery trajectories. Clean demonstrations receive dense stage-progress annotations from a signal-grounded pipeline. From representative demonstrations, Gemini-3.5-Flash[25] generates a shared task template comprising the L3 objective, ordered L2 stages, and their L1 steps. A human reviews this task-level template once, after which its vocabulary and ordering remain fixed across demonstrations of that task. For each demonstration, tactile contact changes identify candidate transition times, supplemented by end-effector motion, gripper state, and visual GEBD cues [59]. After these candidates are merged and deduplicated, the VLM maps each resulting interval to an entry in the task template. It therefore assigns stage semantics to pre-segmented intervals rather than predicting timestamps from the full video. We do not assign monotone labels to complete autonomous or HIL trajectories because they may regress or retry. Instead, tactile contact loss localizes object-drop events, while HIL logs mark correction intervals. These timestamps define local event comparisons. Appendix F reports the task-wise composition of this corpus and provides annotation examples.

Duration-calibrated stage progress. For each stage k, we first compute its mean duration $\bar{d}_k$ across clean demonstrations. Its share of the task progress is

$$w_k = \frac{\bar{d}_k}{\sum_{j=1}^{K} \bar{d}_j}, \qquad \phi_t = \sum_{j<k} w_j + w_k u_t. \tag{6}$$

Here K is the number of stages, and $u_t \in [0,1]$ is the fraction of the current stage completed at frame t. Thus ϕ_t combines the cumulative weights of completed stages with the duration-scaled fraction of the current stage. Weighting stages by their mean durations avoids forcing a brief transition and a long manipulation stage to occupy equal portions of the $[0,1]$ range.

Event-aware pairwise progress learning. We implement $A_\theta(x_a, x_b) \in [-1, 1]$ with a $\pi_{0.5}$-based paired-observation progress architecture [83]. Conditioned on the task prompt, the model jointly encodes observations from two time points, each represented by synchronized RGB images from multiple cameras. A three-layer MLP head then predicts the relative task progress between them. Tactile, kinematic, and event signals construct the offline targets but are not inputs to the progress model. For demonstration pairs, we regress $A_\theta(x_a, x_b)$ toward the target $\phi(x_a) - \phi(x_b)$. We label an observation immediately before an object drop as higher-progress than one immediately after it. For a HIL correction, we label an observation near the end of the intervention as higher-progress than one near its start. Let $\mathcal{D}_{\text{event}}$ contain triples (x_a, x_b, y), where $y = +1$ indicates that x_a is assigned higher progress than x_b, and $y = -1$ indicates the reverse. We optimize

$$\begin{aligned}\mathcal{L}_{\text{prog}} = \mathbb{E}_{\mathcal{D}_{\text{stage}}} \left[A_\theta(x_a, x_b) - (\phi(x_a) - \phi(x_b))\right]^2 \\ + \lambda_{\text{event}} \mathbb{E}_{\mathcal{D}_{\text{event}}} \left[\max(0, m - y A_\theta(x_a, x_b))\right]^2 .\end{aligned} \tag{7}$$

Here $\mathcal{D}_{\text{stage}}$ contains demonstration pairs with dense progress-difference targets, $m > 0$ specifies the minimum signed score $y A_\theta(x_a, x_b)$ required of an event pair, and λ_{event} controls the overall contribution of event comparisons relative to dense demonstration supervision. We randomly reverse event-pair input order during sampling to prevent a fixed input slot from becoming a shortcut.

Advantage-conditioned offline policy learning. For every stored episode, the frozen pairwise progress model produces global progress $\hat{\phi}_t$ and local change $\hat{r}_t$. Samples are assigned to the duration-calibrated stage containing $\hat{\phi}_t$, then ranked only against other samples from that stage:

$$\begin{aligned}&\hat{\phi}_t = \delta_e + A_\theta(x_t, x_0^e), && \hat{r}_t = A_\theta(x_{\min(t+H, T_e-1)}, x_t), \\ &q_k = \text{Quantile}_{1-\rho}\{\hat{r}_i : s_i = k\}, && c_t = \mathbf{1}[\hat{r}_t \geq q_{s_t}].\end{aligned} \tag{8}$$

Here x_0^e is the first observation of episode e, T_e is its number of frames. The offset δ_e is zero for episodes that begin at the nominal task start; staged-recovery episodes use the initialization described in Appendix F. The estimated progress $\hat{\phi}_t$ determines stage index s_t, and q_k is the within-stage threshold. We use the action-chunk horizon $H = 50$ and retain the top $\rho = 0.3$ within each stage as positive, as indicated by $c_t = 1$. We represent this binary indicator as an additional text input. Samples with $c_t = 1$ receive `Advantage: positive`, while the remaining samples receive `Advantage: negative`. The tag is appended to the existing task prompt. For each base policy, ALTER training starts from its pretrained checkpoint. The policy architecture and training objective remain unchanged, including the same flow-matching loss used for supervised task adaptation. At deployment the prompt always uses `Advantage: positive`. Filtering, recovery offsets, short-horizon normalization, and sampling hyperparameters are reported in Appendix F.

5 Experiments

We evaluate $\mathcal{N}_0$-VTLA on the NeoReal real-robot benchmark and the twenty-task simulation suite, test further improvement from deployment data, and close with analyses of the learned representation. Throughout, the base policy of Section 2.1 is initialized from the released $\pi_{0.5}$ weights [58], and the frozen tactile encoder of Section 2.2 is DINOv2 [55].

5.1 Evaluation Protocol

Every comparison in this report is decided by rollout success rate on the target hardware or simulator. Alongside binary success we report a 100-point progress score that awards partial credit for intermediate milestones. Each task is decomposed into a fixed sequence of subtask checkpoints, a rollout earns credit for the deepest checkpoint it reaches, and near-miss behavior on contact-rich tasks stays visible. Representative rubrics appear in Appendix E, with the full set in the companion data report [51].

On NeoReal, each task carries its own trial budget, and outcomes are reported as success rate in percent together with the 100-point stage-rubric progress score. On NeoSim, each policy is trained on 100 demonstrations and evaluated as a success percentage under the simulator's task-completion criterion with a fixed per-task language prompt. Comparisons against the no-tactile base policy are isolated structurally. With

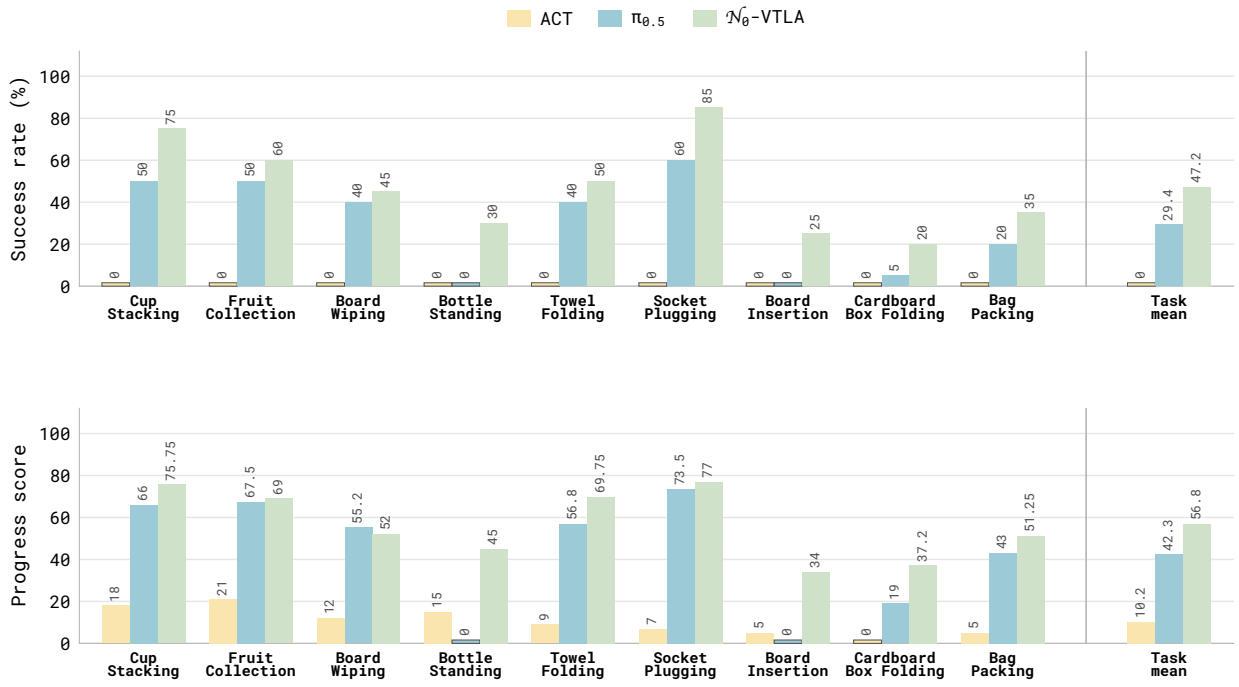

Figure 7 NeoReal benchmark: real-world results. Simulation success rate in the upper panel and the 100-point progress score in the lower panel, on the nine NeoReal contact-rich tasks for ACT, $\pi_{0.5}$, and $\mathcal{N}_0$-VTLA, with exact values printed above each bar and the nine-task means in the rightmost group. Measured zeros appear as thin baseline ticks. $\mathcal{N}_0$-VTLA beats the strongest baseline on every task in success rate and leads the progress-score mean.

its tactile flag disabled, the model reduces to the base policy, so any measured difference in success rate is attributable to the tactile pathway alone.

5.2 Real-World Results: NeoReal

We evaluate $\mathcal{N}_0$-VTLA on nine tasks from NeoReal, a real-world benchmark of fine-grained contact-rich manipulation tasks defined in the $\mathcal{N}_0$-Foundation data report [51]. Tasks run on the corpus's robot-arm platforms under the shared protocol. Figure 11 shows representative rollouts. We deploy the post-trained checkpoints of $\mathcal{N}_0$-VTLA and compare against ACT [93] and $\pi_{0.5}$ [58], both reproduced internally under one aligned evaluation protocol.

Figure 7 plots per-task success rate and progress score. $\mathcal{N}_0$-VTLA beats the strongest baseline on every task in success rate, averaging 47.2% against 29.4% for $\pi_{0.5}$. On the progress score it leads on eight of the nine tasks and in the mean, 56.8 points against 42.3. ACT completes no task and averages 10.2 progress points, stalling in the earliest checkpoints. The margin is clearest on Socket Plugging, the precision outlet insertion, where $\mathcal{N}_0$-VTLA reaches 85% against 60% for $\pi_{0.5}$, and its successful rollouts hold up under daylight lighting shifts and recover from failed insertion attempts. On the long-horizon tasks the progress score separates the systems more sharply than the success rate alone. On Cardboard Box Folding, $\mathcal{N}_0$-VTLA earns 37.2 points of stage credit at a 20% success rate, against 19 points for $\pi_{0.5}$.

5.3 Simulation Results: UniVTAC and NeoSim

The simulation evaluation covers 20 fine-grained contact-rich tasks on the UniVTAC framework, the eight original UniVTAC tasks [18] and the twelve NeoSim tasks of the companion report [51], four single-arm and eight dual-arm. Both suites stress the pre-contact and in-contact regimes where an anticipatory tactile signal

Table 1 UniVTAC benchmark: per-task success rate (%). Closed-loop success on the eight UniVTAC tasks for $\mathcal{N}_0$-VTLA and external baselines, all under one aligned protocol. ACT (Vision Only) drops the tactile stream; ACT + UniVTAC and VITaL [29] add it. Best per task in **bold**.

Method	Lift Bottle	Pull-out Key	Lift Can	Put Bottle	Insert Hole	Insert HDMI	Insert Tube	Grasp Classify	Avg
ACT (Vision Only)	42	28	20	28	19	15	45	50	30.9
ACT + UniVTAC	71	46	29	31	24	28	56	99	48.0
VITaL	72	47	8	32	25	6	34	**100**	40.5
$\pi_{0.5}$	**100**	35	6	34	25	8	74	49	41.4
StarVLA-α	32	51	65	**88**	52	24	69	68	56.1
InternVLA-A1	58	66	**90**	37	93	12	95	86	67.1
Xiaomi-Robotics-0	21	80	13	12	**96**	**69**	98	45	54.3
GigaWorld-Policy	38	32	0	21	12	0	9	20	16.5
$\mathcal{N}_0$-VTLA	99	**99**	88	60	95	25	**99**	**100**	**83.1**

Table 2 NeoSim benchmark: per-task success rate (%). Closed-loop success on the twelve NeoSim tasks of the companion report [51], split into four single-arm and eight dual-arm tasks, for $\mathcal{N}_0$-VTLA and external baselines under one aligned protocol. Best per task in **bold**; group and suite means in the shaded rows.

Task	$\pi_{0.5}$	StarVLA-α	InternVLA-A1	Xiaomi-Rob.-0	GigaWorld-P.	**$\mathcal{N}_0$-VTLA**
Single-arm tasks						
Pour Ball	92	32	47	52	0	**96**
Unplug and Plug Charger	23	0	0	5	3	**30**
Plug USB	74	72	36	62	34	**81**
Grasp Chip	86	**93**	12	49	71	88
Single-arm mean	68.8	49.3	23.8	42.0	27.0	**73.8**
Dual-arm tasks						
Insert Screw	**26**	8	0	0	0	24
Place Gears	18	0	0	0	0	**29**
Unstack Bowl	3	**17**	0	0	0	**17**
Cup Handover	13	**25**	0	0	0	22
Stack Cups	22	0	0	0	12	**23**
Stack Plates	**96**	0	0	0	0	94
Stack Bowls	93	0	8	42	0	**96**
Unstack Cup	3	31	0	**71**	9	10
Dual-arm mean	34.3	10.1	1.0	14.1	2.6	**39.4**
NeoSim mean	45.8	23.2	8.6	23.4	10.8	**50.8**

should matter most. We report per-task success rate in percent for $\mathcal{N}_0$-VTLA alongside external baselines, namely $\pi_{0.5}$ [58], StarVLA-α [82], InternVLA-A1 [14], Xiaomi-Robotics-0 [15], and GigaWorld-Policy [81], all evaluated under one aligned protocol on the same task configurations.

Table 1 reports the eight UniVTAC tasks and Table 2 the twelve NeoSim tasks. On UniVTAC, $\mathcal{N}_0$-VTLA averages 83.1%, ahead of the strongest external baseline, InternVLA-A1 at 67.1%, with near-saturating success on several insertion tasks and 100% on grasp-and-classify. On NeoSim the twelve tasks are harder for every method; the specialist baselines fall to between 8.6% and 23.4%, while $\mathcal{N}_0$-VTLA holds 50.8% against 45.8% for $\pi_{0.5}$. The gap between arms is stark: on the four single-arm tasks $\mathcal{N}_0$-VTLA averages

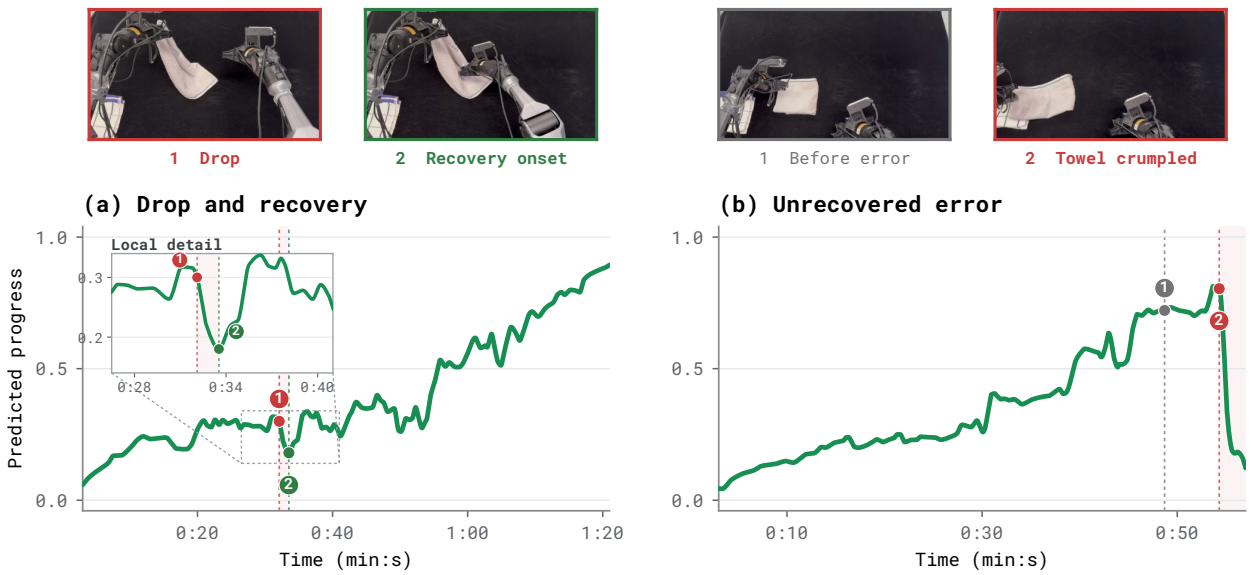

Figure 8 Predicted progress on imperfect Towel Folding deployments. Predictions from our pairwise progress model on two held-out trajectories containing (a) a corrected towel drop and (b) an unrecovered crumpling error. Numbered markers indicate the synchronized frames, and pale red regions denote degraded execution. The inset in (a) enlarges the region around recovery onset.

73.8%, but the eight dual-arm tasks halve it to 39.4% and collapse the specialists into the single digits, InternVLA-A1 to 1.0%. Averaged over all twenty tasks, $\mathcal{N}_0$-VTLA reaches 63.8% against 44.0% for $\pi_{0.5}$, its strongest overall baseline. These insertion-heavy and bimanual regimes are the contact-decided setting the latent tactile pathway targets.

5.4 Offline Policy Improvement with ALTER

We evaluate ALTER by training task-specific policies from pretrained VLA and VTLA checkpoints. The evaluation first characterizes the pairwise progress model's response to local execution regressions and then compares the policies on real-robot tasks.

Progress response on imperfect rollouts. Figure 8 shows the predicted global progress on two held-out, imperfect Towel Folding trajectories. In the first trajectory, predicted progress drops around the towel drop, reaches a local minimum near recovery onset, and then resumes its upward trend as the robot recovers. In the second, predicted progress falls sharply after the flattened towel becomes crumpled during transport and remains low through episode termination because no recovery follows. In these examples, predicted progress therefore decreases at both observed degradations and rises again only when recovery follows. Appendix F provides the corresponding qualitative check on the other two tasks.

Policy comparison. We evaluate five variants on Towel Folding, Bag Packing, and Cardboard Box Folding. $\pi_{0.5}$-*SFT* and $\mathcal{N}_0$-*VTLA-SFT* are task-adapted using the demonstration subset. $\pi_{0.5}$+*ALTER* and $\mathcal{N}_0$-*VTLA*+*ALTER* apply the complete offline policy-learning procedure to the two base policies. $\pi_{0.5}+\chi_0$ replaces our progress-model supervision with the Stage Advantage supervision of χ_0 [83]. Each offline policy-learning variant starts from its corresponding pretrained model.

Across the three tasks, ALTER with a $\pi_{0.5}$ backbone obtains success rates of 90%, 75%, and 60%; with $\mathcal{N}_0$-VTLA, it reaches 95%, 80%, and 75%. Thus, the VTLA backbone retains a consistent final advantage under the same offline RL procedure. The $\mathcal{N}_0$-VTLA policy trained with ALTER completes the Towel Folding, Bag Packing, and Cardboard Box Folding tasks at throughputs of 43, 15, and 30 successful executions per hour, respectively. The corresponding comparison appears in Appendix F.

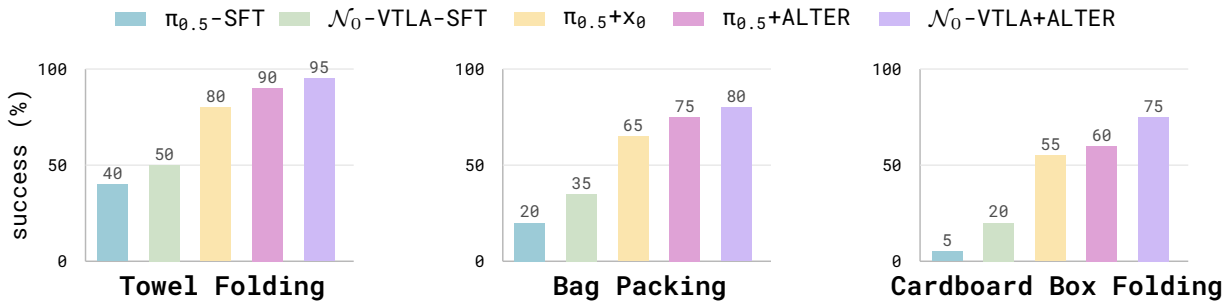

Figure 9 Real-robot success rates under supervised and offline policy learning. All three tasks use the same 0–100% vertical scale, and all five variants report measured rollouts.

5.5 Representation Analyses

Our final analyses probe the learned tactile representation directly. At the end of Stage 1 pretraining we measure three things. The first is held-out contrastive retrieval, the accuracy form of the Stage 1 InfoNCE objective. Over 378 held-out queries, the predicted z retrieves its matching future-tactile target z^* of Eq. 2 top-1 in 92.3% of cases within a pool of roughly 32 candidates, where chance is 3.2%. These are within-pool rates, not full-corpus retrieval. The second is a control that separates anticipation from autocorrelation. Ranking the same candidates by the current tactile encoding g alone, the predictor's own input, reaches 57% top-1, and 40% in a 128-candidate pool where the predictor holds 81%. The latent therefore carries future contact information beyond its input, and the margin grows as the pool gets harder. The third is a perturbation probe. Swapping the tactile input moves z by roughly 0.9 in centered-cosine distance while swapping the RGB views and prompt together moves it by no more than about 0.2, and the tactile-to-vision-language sensitivity ratio settles at 4.3 by the end of Stage 1 and measures about 1.4 after end-to-end joint training.

The predictor is grounded in touch, and it anticipates. A latent that merely copied its tactile input would match the current-touch baseline at 57%. The measured 92.3% sits far above it, and the gap widens as the candidate pool grows. The perturbation probe argues against a vision–language shortcut. Swapping that context barely moves z, and the latent stays tactile-leaning even after joint training. The representation entering joint training is therefore a real, anticipatory one.

6 Findings

Touch changes manipulation where contact decides it. Four findings collect the evidence.

Touch turns insertion from a one-shot visual commitment into a contact-aware retry loop. The two hardest real-robot insertions show the clearest margins of the campaign. Socket Plugging, inserting a plug into an outlet, reaches 85% against 60% for $\pi_{0.5}$, and Board Insertion, seating an expansion card, reaches 25% on a task where ACT and $\pi_{0.5}$ both fail outright at 0%. Representative rollouts reveal the behavioral difference behind these gains. Once visual alignment suggests that insertion is possible, $\pi_{0.5}$ commits to the downward motion. When the plug or card meets the rim instead of entering the slot, it continues the attempt and fails. $\mathcal{N}_0$-VTLA instead responds to the unexpected contact by lifting the arm, realigning, and attempting the insertion again. The pattern repeats in simulation, where Insert Hole and Insert Tube reach 95% and 99%. The tactile pathway therefore changes insertion from a single visually planned motion into a closed-loop process that can detect a blocked attempt and recover from it.

Touch enables fine control of gripper force through aperture adjustment. Grip force rather than placement decides a second family of tasks. In the real-robot Bottle Standing task, the manipulated object

is an empty, compliant plastic bottle. $\pi_{0.5}$ closes the gripper too far around the bottle mouth, pinching it firmly enough to lift the entire bottle instead of leaving it on the supporting surface. $\mathcal{N}_0$-VTLA instead makes small, continuous adjustments to the gripper aperture, maintaining sufficient contact to control the bottle without gripping hard enough to lift it, and consequently stabilizes the bottle upright. Although the tactile sensor does not measure force directly, its deformation signal is strongly coupled to contact force and provides a practical feedback signal for regulating grip. This behavior is consistent with the broader results. Bottle Standing reaches 30% where both real-robot baselines remain at 0%, while Lift Can and Grasp Chip reach 88% in simulation.

Touch changes the action exactly when contact decides the outcome. Figure 10 probes the mechanism directly. At the same observation and under identical sampling noise, the predicted action path with touch diverges from its touch-removed counterpart at the contact-critical moments of firm contact and grasp, and coincides with it in free space. Because everything else is held fixed, the comparison isolates the tactile pathway's contribution to the sampled action. In the probed episode it is silent in free space and decisive at contact.

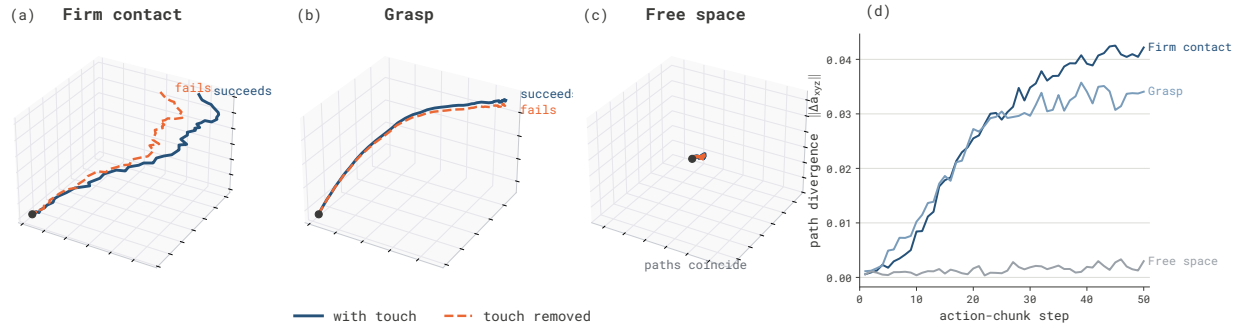

Figure 10 A counterfactual probe of what touch changes. At the same observation and under identical sampling noise, the policy's predicted end-effector path with touch is compared with its touch-removed counterfactual at three moments of one episode. At firm contact and grasp the paths separate, and the with-touch path is the one that succeeds. In free space, drawn at the same spatial scale as (a), the two coincide. Panel (d) plots the per-step divergence of the two paths over the 50-step chunk: it grows through the contact-critical chunks and stays near zero in free space. Axes are normalized action units.

Stronger tactile pretraining remains advantageous under ALTER. Under clean-demonstration SFT, $\mathcal{N}_0$-VTLA leads $\pi_{0.5}$ by 10, 15, and 15 points on Towel Folding, Bag Packing, and Cardboard Box Folding. When each backbone is trained with ALTER from its corresponding pretrained checkpoint, $\mathcal{N}_0$-VTLA reaches 95%, 80%, and 75%, compared with 90%, 75%, and 60% for $\pi_{0.5}$. The advantage associated with the stronger VTLA checkpoint therefore persists under the same offline RL procedure.

7 Related Work

Vision–language–action policies. Modern vision–language–action (VLA) policies build on action-chunk and generative visuomotor architectures established by ACT [93] and Diffusion Policy [23]. ACT predicts temporally coherent action chunks, whereas Diffusion Policy models multimodal action sequences through iterative denoising. RT-1 [11] established transformer policies for real-world control at scale, and RT-2 [10] subsequently connected internet-pretrained vision–language representations to robot control, while RT-X [54] and Octo [53] scaled generalist policies across heterogeneous robot datasets. Subsequent VLA systems have diversified along two broad axes: their action-generation mechanisms and their strategies for scaling, transfer, and deployment. OpenVLA [37] provides an open autoregressive backbone, while π_0 [9] and $\pi_{0.5}$ [58] generate continuous action chunks through flow matching. Autoregressive action tokens reuse the language-model interface and scale naturally with pretrained backbones, whereas continuous generative heads model low-level commands directly in continuous space and can capture multiple distinct, valid action sequences.

Beyond action generation, UniVLA [12] and Qwen-VLA [67] extend transfer across tasks, environments, and embodiments. StarVLA [61] emphasizes modularity, MiMo-Embodied [32] targets cross-embodiment scaling, and Pragmatic VLA [72] focuses on practical deployment. Recent foundation-scale systems push data and model size further: Xiaomi-Robotics-1 [74] pretrains on more than 100,000 hours of real-world trajectories, its U0 variant [41] couples action synthesis with a world foundation model, CronusVLA [40] aggregates multi-frame context, InternVLA-A1.5 [47] unifies understanding, latent foresight, and action, and GR-3 [16], GR00T N1 [8], and SmolVLA [60] span large generalist to affordable efficient policies. Benchmark suites such as VLABench [89] and LIBERO-Plus [27] track this progress, probing long-horizon reasoning and robustness under controlled perturbations. Across these systems, generalization is pursued either by expanding heterogeneous pretraining mixtures or by learning task and embodiment abstractions that can be reused during adaptation. Together, these lines combine semantic grounding, cross-robot data scaling, and generative continuous control. Their observation spaces, however, remain centered on cameras, language, and robot state, leaving contact and slip only indirectly observable. $\mathcal{N}_0$-VTLA retains the language-conditioned, flow-matching $\pi_{0.5}$ backbone but adds a dedicated high-resolution tactile pathway.

Tactile-conditioned and predictive VTLA policies. Work toward tactile-conditioned vision–language–action control combines transferable tactile representations and scalable data collection with policies that either react to measured touch or predict how contact will evolve. Touch and Go [79] and Visuo-Tactile Transformers [21] learn joint visual–tactile features. VITaL [29], Sparsh [33], AnyTouch [28], and UniTouch [80] emphasize transfer across tasks or sensors, and FTP-1 [85] scales cross-sensor transfer to a generalist tactile policy pretrained on thousands of hours of tactile data. FreeTacMan [70], exUMI [76], ViTaMIn [44], Touch in the Wild [97], and UniVTAC [18] use robot-free collection or simulation to reduce contact-rich data costs. Reactive Diffusion Policy [77] uses touch for fast feedback, 3D-ViTac [35] performs spatial fusion, TouchGuide [91] provides inference-time steering, and Multi-Modal Policy Consensus [19] fuses controllers, and T-Rex [52] pairs a variable-rate architecture with high-frequency touch for tactile-reactive dexterous control. TLA [31], Tactile-VLA [36], VLA-Touch [7], and OmniVTLA [22] further integrate observed touch into language-conditioned control, while Neural Feels [63] targets fine-grained state estimation. Seeing Touch from Motion [75] reads fine-grained contact states from tactile motion correlation, fusing touch and vision in a modality-aware mixture-of-transformers policy. Force-aware variants condition on measured or estimated force: ForceVLA [84] adds a force-aware mixture-of-experts, and TA-VLA [90] a torque-aware interface for contact-rich control. Together, they establish the value of current touch for closed-loop correction and grounding. Predictive methods complement this reactive signal by estimating contact before it unfolds: Imagine2Touch [3] predicts local tactile readings ahead of contact, and TTP [88] transfers a future-tactile expert to robot policies. HapticVLA [30] likewise estimates rather than measures touch, though its target is the current signal: a tactile token inferred from vision and robot state replaces the sensor at deployment. Visuo-Tactile World Models [34], OmniVTA [94], Tactile-WAM [71], Dream-Tac [45], VT-WAM [65], TacForeSight [87], and ContactWorld [92] instead forecast richer dynamics, sensory futures, and actions. A parallel predictive line forecasts future observations rather than latents: video-generation policies such as UniPi [26], GR-1 [69], and GR-2 [17] synthesize pixel futures to guide action, and Unified World Models [96] couple video and action diffusion, and World Guidance [62] models the world in a learned condition space rather than in pixels to guide action generation. Against this backdrop, $\mathcal{N}_0$-VTLA occupies a lighter point in the design space. Rather than training a tactile-specific encoder or decoding a full sensory trajectory, it repurposes frozen DINOv2 features [55], retains measured touch, and uses vision and language to predict its net change over the action horizon. The resulting compact latent conditions the action expert directly. This latent-target formulation is related to I-JEPA [1], V-JEPA [5], V-JEPA 2 [2], LeJEPA [4], and LeWorldModel [49]. Its distinction lies in the target, placement, training recipe, and scale within a shared language-conditioned VLA.

Offline RL and VLA policy improvement from deployment data. Deployment exposes distribution shifts and long-tail failures absent from static demonstrations, while human interventions provide corrective data. RECAP [57] learns from demonstrations, autonomous experience, and interventions through advantage-conditioned policy extraction. SOP [56] scales collection and learning across robot fleets. LWD [68] uses value-guided updates for flow-based VLAs. VLA-RL [46] performs trajectory-level RL with segment-derived process rewards. A complementary line converts sparse outcomes into dense process supervision.

Vision–language models can serve as in-context value estimators [48], while Robo-Dopamine [64] learns step-aware, multi-view rewards. SARM [20] models stages and within-stage progress for filtering and reweighting variable-duration demonstrations. ARM [50] labels relative progression, regression, or stagnation. Learned progress or value then filters demonstrations in GR-RL [42], weights flow matching in ProgVLA [38], guides actions in ProgressVLA [78], and supports offline-to-online adaptation in Robo-ValueRL [73]. Closest to our setting, χ_0 [83] predicts stage-aware relative progress from paired observations, while RECAP [57] conditions VLA training on binarized advantages. χ_0 derives its progress targets by spacing stages uniformly and using elapsed time. ALTER instead weights stages by demonstrated duration and adds local comparison supervision around tactile-detected object-drop events and logged HIL corrections for advantage-conditioned offline policy learning.

8 Conclusion

We presented $\mathcal{N}_0$-VTLA, a vision–tactile–language–action foundation model that predicts outcomes instead of reacting to them. For perception, *latent tactile tokens* predict contact. A frozen visual backbone with a trainable projection turns the gripper's contact-difference images from our self-developed visuo-tactile sensor into tokens, and a lightweight predictor compresses these tokens, together with the vision–language context, into a latent z that conditions a flow-matching action expert directly, never entering the vision–language prefix, so that touch is treated as a prediction target rather than as additional observation context. For policy improvement, ALTER formulates learning from deployment corpus as advantage-conditioned offline RL. Its pairwise progress model estimates global task phase and local execution change, which are converted into stage-relative advantage conditions for offline policy learning. A three-stage recipe brings this newly initialized pathway online without destabilizing the pretrained $\pi_{0.5}$ backbone. Stage 1 grounds the predictor against a future-tactile target, Stage 2 freezes the tactile perception stack and masks the vision–language pathway to align the resulting latents with the action expert, and Stage 3 trains the full policy jointly. The recipe rests on a canonical cross-embodiment action space and NeoData, a large-scale curated multi-platform corpus. Representation-level evidence already supports the design. After Stage 1, the latent tokens retrieve their matching future-tactile target at 92.3% top-1 accuracy, far above the 3.2% chance level. The closed-loop results point the same way. $\mathcal{N}_0$-VTLA attains the highest average on the simulation suite by a margin of more than nineteen points over the strongest baseline, and goes unbeaten across the nine NeoReal real-robot tasks.

Future work. Two directions follow directly from this report. First, the free-latent and supervised variants defined here are two points in a broader design space for tactile representation learning, and we see the predictive-latent framing, conditioning action generation on a compact estimate of the net contact change over the chunk horizon rather than on raw sensory tokens, as a direction worth pursuing beyond the specific predictor architecture used in this report. Second, ALTER can be extended to a broader range of tasks to study its effectiveness across different manipulation settings.

Contributors

Pretraining. Heng Zhou, Silong Dai, Yutao Fan, Yiran Qin, Shunlin Lu.
Offline RL. Yutao Fan, Longjie Su, Heng Zhou, Yiming Wu, Yiran Qin, Shunlin Lu.
Post-Training (Simulation). Heng Zhou, Bruno N.Y. Chen, Zhemeng Zhang, Yiran Qin, Shunlin Lu.
Post-Training (Real Robot). Jiongwei Lu, Silong Dai, Heng Zhou, Yutao Fan, Zhemeng Zhang, Shengqi Xu, Boyu Mi, Bruno N.Y. Chen, Li Kang, Yanjun Li.
Data processing. Yutao Fan, Heng Zhou, Rui Li, Xiufeng Song, Tianyu Yang, Wenjie Zhou, Yifan Wang, Yiming Wu, Xin Wang, Bruno N.Y. Chen.
Academic Supervision. Ziyi Ye, Guoxiang Dong, Xiaosong Jia, Wenming Chen.
Project Lead. Yiran Qin, Shunlin Lu, Shihao Zhao, Daoguo Dong, Zuxuan Wu.

References

[1] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. *CVPR*, 2023.

[2] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[3] Abdallah Ayad, Adrian Röfer, Nick Heppert, and Abhinav Valada. Imagine2touch: Predictive tactile sensing for robotic manipulation using efficient low-dimensional signals. *arXiv preprint arXiv:2405.01192*, 2024.

[4] Randall Balestriero and Yann LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.

[5] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *TMLR*, 2024.

[6] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. PaliGemma: A versatile 3B VLM for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[7] Jianxin Bi, Kevin Yuchen Ma, Ce Hao, Mike Zheng Shou, and Harold Soh. VLA-Touch: Enhancing vision-language-action models with dual-level tactile feedback. *RA-L*, 2026.

[8] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[9] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *RSS*, 2025.

[10] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *CoRL*, 2023.

[11] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, et al. RT-1: Robotics transformer for real-world control at scale. *RSS*, 2023.

[12] Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. UniVLA: Learning to act anywhere with task-centric latent actions. *RSS*, 2025.

[13] Remi Cadene, Simon Aliberts, Francesco Capuano, Michel Aractingi, Adil Zouitine, Pepijn Kooijmans, Jade Choghari, Martino Russi, Caroline Pascal, Steven Palma, et al. LeRobot: An open-source library for end-to-end robot learning. *ICLR*, 2026.

[14] Junhao Cai, Zetao Cai, Jiafei Cao, Yilun Chen, Zeyu He, Lei Jiang, Hang Li, Hengjie Li, Yang Li, Yufei Liu, et al. InternVLA-A1: Unifying understanding, generation and action for robotic manipulation. *arXiv preprint arXiv:2601.02456*, 2026.

[15] Rui Cai, Jun Guo, Xinze He, Piaopiao Jin, Jie Li, Bingxuan Lin, Futeng Liu, Wei Liu, Fei Ma, Kun Ma, et al. Xiaomi-Robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv preprint arXiv:2602.12684*, 2026.

[16] Chilam Cheang, Sijin Chen, Zhongren Cui, Yingdong Hu, Liqun Huang, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Xiao Ma, et al. GR-3 technical report. *arXiv preprint arXiv:2507.15493*, 2025.

[17] Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, et al. GR-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

[18] Baijun Chen, Weijie Wan, Tianxing Chen, Xianda Guo, Congsheng Xu, Yuanyang Qi, Haojie Zhang, Longyan Wu, Tianling Xu, Zixuan Li, et al. UniVTAC: A unified simulation platform for visuo-tactile manipulation data generation, learning, and benchmarking. *arXiv preprint arXiv:2602.10093*, 2026.

[19] Haonan Chen, Jiaming Xu, Hongyu Chen, Kaiwen Hong, Binghao Huang, Chaoqi Liu, Jiayuan Mao, Yunzhu Li, Yilun Du, and Katherine Driggs-Campbell. Multi-modal manipulation via multi-modal policy consensus. *ICRA*, 2026.

[20] Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Yide Shentu, and Philipp Wu. SARM: Stage-aware reward modeling for long horizon robot manipulation. *ICLR*, 2026.

[21] Yizhou Chen, Andrea Sipos, Mark Van der Merwe, and Nima Fazeli. Visuo-tactile transformers for manipulation. *CoRL*, 2022.

[22] Zhengxue Cheng, Yiqian Zhang, Anni Tang, Keyu Wang, Wenkang Zhang, Haoyu Li, Hengdi Zhang, and Li Song. OmniVTLA: Vision-tactile-language-action models with semantic-aligned tactile sensing. *RA-L*, 2026.

[23] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *RSS*, 2023.

[24] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *RSS*, 2024.

[25] Google DeepMind. Gemini 3.5 flash. https://deepmind.google/models/gemini/flash/, 2025.

[26] Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *NeurIPS*, 2023.

[27] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *CVPR*, 2026.

[28] Ruoxuan Feng, Jiangyu Hu, Wenke Xia, Tianci Gao, Ao Shen, Yuhao Sun, Bin Fang, and Di Hu. AnyTouch: Learning unified static-dynamic representation across multiple visuo-tactile sensors. *ICLR*, 2025.

[29] Abraham George, Selam Gano, Pranav Katragadda, and Amir Barati Farimani. VITaL pretraining: Visuo-tactile pretraining for tactile and non-tactile manipulation policies. *ICRA*, 2025.

[30] Konstantin Gubernatorov, Mikhail Sannikov, Ilya Mikhalchuk, Egor Kuznetsov, Makar Artemov, Ogunwoye Faith Ouwatobi, Marcelino Fernando, Artem Asanov, Ziang Guo, and Dzmitry Tsetserukou. HapticVLA: Contact-rich manipulation via vision-language-action model without inference-time tactile sensing. *arXiv preprint arXiv:2603.15257*, 2026.

[31] Peng Hao, Chaofan Zhang, Dingzhe Li, Xiaoge Cao, Xiaoshuai Hao, Shaowei Cui, and Shuo Wang. TLA: Tactile-language-action model for contact-rich manipulation. *Robot Learning*, 2026.

[32] Xiaoshuai Hao, Lei Zhou, Zhijian Huang, Zhiwen Hou, Yingbo Tang, Lingfeng Zhang, Guang Li, Zheng Lu, Shuhuai Ren, Xianhui Meng, et al. MiMo-Embodied: X-embodied foundation model technical report. *arXiv preprint arXiv:2511.16518*, 2025.

[33] Carolina Higuera, Akash Sharma, Chaithanya Krishna Bodduluri, Taosha Fan, Patrick Lancaster, Mrinal Kalakrishnan, Michael Kaess, Byron Boots, Mike Lambeta, Tingfan Wu, et al. Sparsh: Self-supervised touch representations for vision-based tactile sensing. *CoRL*, 2024.

[34] Carolina Higuera, Sergio Arnaud, Byron Boots, Mustafa Mukadam, Francois Robert Hogan, and Franziska Meier. Visuo-tactile world models. *arXiv preprint arXiv:2602.06001*, 2026.

[35] Binghao Huang, Yixuan Wang, Xinyi Yang, Yiyue Luo, and Yunzhu Li. 3D-ViTac: Learning fine-grained manipulation with visuo-tactile sensing. *CoRL*, 2024.

[36] Jialei Huang, Shuo Wang, Fanqi Lin, Yihang Hu, Chuan Wen, and Yang Gao. Tactile-VLA: Unlocking vision-language-action model's physical knowledge for tactile generalization. *arXiv preprint arXiv:2507.09160*, 2025.

[37] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. *CoRL*, 2024.

[38] Seungsu Kim, Jinyoung Choi, Seungmin Baek, and Jean-Michel Renders. ProgVLA: Progress-aware robot manipulation skill learning. *arXiv preprint arXiv:2605.28231*, 2026.

[39] Mike Lambeta, Po-Wei Chou, Stephen Tian, Brian Yang, Benjamin Maloon, Victoria Rose Most, Dave Stroud, Raymond Santos, Ahmad Byagowi, Gregg Kammerer, et al. DIGIT: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation. *RA-L*, 2020.

[40] Hao Li, Shuai Yang, Yilun Chen, Xinyi Chen, Xiaoda Yang, Yang Tian, Hanqing Wang, Tai Wang, Dahua Lin, Feng Zhao, et al. CronusVLA: Towards efficient and robust manipulation via multi-frame vision-language-action modeling. *AAAI*, 2026.

[41] Xinghang Li, Jun Guo, Qiwei Li, Long Qian, Hang Lai, Yueze Wang, Hongyu Yan, Jiahang Cao, Xi Chen, Jingen Qu, et al. Xiaomi-Robotics-U0: Unified embodied synthesis with world foundation model. *arXiv preprint arXiv:2607.11643*, 2026.

[42] Yunfei Li, Xiao Ma, Jiafeng Xu, Yu Cui, Zhongren Cui, Zhigang Han, Liqun Huang, Tao Kong, Yuxiao Liu, Hao Niu, et al. GR-RL: Going dexterous and precise for long-horizon robotic manipulation. *arXiv preprint arXiv:2512.01801*, 2025.

[43] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *ICLR*, 2023.

[44] Fangchen Liu, Chuanyu Li, Yihua Qin, Jing Xu, Pieter Abbeel, and Rui Chen. ViTaMIn: Learning contact-rich tasks through robot-free visuo-tactile manipulation interface. *arXiv preprint arXiv:2504.06156*, 2025.

[45] Yunfan Lou, Yifan Ye, Yankai Fu, Jun Cen, Xiaowei Chi, Yaoxu Lyu, Peidong Jia, Sirui Han, Zhihe Lu, and Shanghang Zhang. Dream-Tac: A unified tactile world action model for contact-rich robot manipulation. *arXiv preprint arXiv:2606.08737*, 2026.

[46] Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. VLA-RL: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint arXiv:2505.18719*, 2025.

[47] Haoxiang Ma, Junhao Cai, Xiaoxu Xu, Hao Li, Yuyin Yang, Yang Tian, Jiafei Cao, Hongrui Zhu, Zherui Qiu, Zhaxizhuoma, et al. InternVLA-A1.5: Unifying understanding, latent foresight, and action for compositional generalization. *arXiv preprint arXiv:2607.04988*, 2026.

[48] Yecheng Jason Ma, Joey Hejna, Ayzaan Wahid, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, et al. Vision language models are in-context value learners. *ICLR*, 2025.

[49] Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

[50] Yiming Mao, Zixi Yu, Weixin Mao, Yinhao Li, Qirui Hu, Zihan Lan, Minzhao Zhu, and Hua Chen. ARM: Advantage reward modeling for long-horizon manipulation. *arXiv preprint arXiv:2604.03037*, 2026.

[51] NeoteAI Team and Fudan TEAI Team. $\mathcal{N}_0$-foundation: Towards the age of tactile intelligence. https://research.neoteai.com/n0-foundation/, 2026.

[52] Dantong Niu, Zhuoyang Liu, Zekai Wang, Boning Shao, Zhao-Heng Yin, Anirudh Pai, Yuvan Sharma, Stefano Saravalle, Ruijie Zheng, Jing Wang, et al. T-Rex: Tactile-reactive dexterous manipulation. *arXiv preprint arXiv:2606.17055*, 2026.

[53] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, et al. Octo: An open-source generalist robot policy. *RSS*, 2024.

[54] Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, et al. Open X-embodiment: Robotic learning datasets and RT-X models. *ICRA*, 2024.

[55] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024.

[56] Mingjie Pan, Siyuan Feng, Qinglin Zhang, Xinchen Li, Jianheng Song, Chendi Qu, Yi Wang, Chuankang Li, Ziyu Xiong, Zhi Chen, et al. SOP: A scalable online post-training system for vision-language-action models. *arXiv preprint arXiv:2601.03044*, 2026.

[57] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, et al. $\pi^*_{0.6}$: a VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

[58] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *CoRL*, 2025.

[59] Mike Zheng Shou, Stan Weixian Lei, Weiyao Wang, Deepti Ghadiyaram, and Matt Feiszli. Generic event boundary detection: A benchmark for event segmentation. *ICCV*, 2021.

[60] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[61] StarVLA Community. StarVLA: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

[62] Yue Su, Sijin Chen, Haixin Shi, Mingyu Liu, Zhengshen Zhang, Ningyuan Huang, Weiheng Zhong, Zhengbang Zhu, Yuxiao Liu, and Xihui Liu. World guidance: World modeling in condition space for action generation. *arXiv preprint arXiv:2602.22010*, 2026.

[63] Sudharshan Suresh, Haozhi Qi, Tingfan Wu, Taosha Fan, Luis Pineda, Mike Lambeta, Jitendra Malik, Mrinal Kalakrishnan, Roberto Calandra, Michael Kaess, et al. NeuralFeels with neural fields: Visuotactile perception for in-hand manipulation. *Sci. Robot.*, 2024.

[64] Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Yuheng Ji, Cheng Chi, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, et al. Robo-Dopamine: General process reward modeling for high-precision robotic manipulation. *CVPR*, 2026.

[65] Shuai Tian, Yupeng Zheng, Yuhang Zheng, Songen Gu, Yujie Zang, Yuxing Qin, Weize Li, Haoran Li, Wenchao Ding, and Dongbin Zhao. VT-WAM: Visual-tactile world action model for contact-rich manipulation. *arXiv preprint arXiv:2607.02503*, 2026.

[66] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[67] Qiuyue Wang, Mingsheng Li, Jian Guan, Jinhui Ye, Sicheng Xie, Yitao Liu, Junhao Chen, Zhixuan Liang, Jie Zhang, Xintong Hu, et al. Qwen-VLA: Unifying vision-language-action modeling across tasks, environments, and robot embodiments. *arXiv preprint arXiv:2605.30280*, 2026.

[68] Yi Wang, Xinchen Li, Pengwei Xie, Pu Yang, Buqing Nie, Yunuo Cai, Qinglin Zhang, Chendi Qu, Jeffrey Wu, Jianheng Song, et al. Learning while deploying: Fleet-scale reinforcement learning for generalist robot policies. *arXiv preprint arXiv:2605.00416*, 2026.

[69] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. *ICLR*, 2024.

[70] Longyan Wu, Checheng Yu, Jieji Ren, Li Chen, Yufei Jiang, Ran Huang, Guoying Gu, and Hongyang Li. FreeTacMan: Robot-free visuo-tactile data collection system for contact-rich manipulation. *ICRA*, 2026.

[71] Siyu Wu, Linjing You, Junjie Zhu, Yaozu Liu, Changhao Zhang, Jian Liu, Weiqiang Wang, Qi Li, Jituo Li, and Hengshuang Zhao. Tactile-WAM: Touch-aware world action model with tactile asymmetric attention. *arXiv preprint arXiv:2606.26663*, 2026.

[72] Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, et al. A pragmatic VLA foundation model. *arXiv preprint arXiv:2601.18692*, 2026.

[73] Wenke Xia, Pei Ren, Wenbo Yu, Yizhuo Zhang, Jifan Li, Yixue Zhang, Yinuo Zhao, Qingyang Gao, Jianlong Fu, Jian Tang, et al. Robo-ValueRL: Reliable value estimation for offline-to-online reinforcement learning. *arXiv preprint arXiv:2607.09866*, 2026.

[74] Xiaomi Robotics Team, Jun Guo, Piaopiao Jin, Jason Li, Peiyan Li, Yingyan Li, Futeng Liu, Wanli Peng, Optimus Qin, Yifei Su, Nan Sun, et al. Xiaomi-Robotics-1: Scaling vision-language-action models with over 100K hours of real-world trajectories. *arXiv preprint arXiv:2607.15330*, 2026.

[75] Shengqi Xu, Guojin Zhong, Yang Liu, Fanjie Wang, Hu Luo, Hanyu Zhou, Weiyao Zhang, Ziyi Ye, Zuxuan Wu, and Yu-Gang Jiang. Seeing touch from motion: A unified modality-aware visuo-tactile policy with tactile motion correlation. *ECCV*, 2026.

[76] Yue Xu, Litao Wei, Pengyu An, Qingyu Zhang, and Yong-Lu Li. exUMI: Extensible robot teaching system with action-aware task-agnostic tactile representation. *CoRL*, 2025.

[77] Han Xue, Jieji Ren, Wendi Chen, Gu Zhang, Yuan Fang, Guoying Gu, Huazhe Xu, and Cewu Lu. Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation. *RSS*, 2025.

[78] Hongyu Yan, Qiwei Li, Jiaolong Yang, and Yadong Mu. ProgressVLA: Progress-guided diffusion policy for vision-language robotic manipulation. *arXiv preprint arXiv:2603.27670*, 2026.

[79] Fengyu Yang, Chenyang Ma, Jiacheng Zhang, Jing Zhu, Wenzhen Yuan, and Andrew Owens. Touch and go: Learning from human-collected vision and touch. *NeurIPS*, 2022.

[80] Fengyu Yang, Chao Feng, Ziyang Chen, Hyoungseob Park, Daniel Wang, Yiming Dou, Ziyao Zeng, Xien Chen, Rit Gangopadhyay, Andrew Owens, et al. Binding touch to everything: Learning unified multimodal tactile representations. *CVPR*, 2024.

[81] Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, et al. GigaWorld-Policy: An efficient action-centered world–action model. *arXiv preprint arXiv:2603.17240*, 2026.

[82] Jinhui Ye, Ning Gao, Senqiao Yang, Jinliang Zheng, Zixuan Wang, Yuxin Chen, Pengguang Chen, Yilun Chen, Shu Liu, and Jiaya Jia. StarVLA-α: Reducing complexity in vision-language-action systems. *arXiv preprint arXiv:2604.11757*, 2026.

[83] Checheng Yu, Chonghao Sima, Gangcheng Jiang, Hai Zhang, Haoguang Mai, Hongyang Li, Huijie Wang, Jin Chen, Kaiyang Wu, Li Chen, et al. χ_0: Resource-aware robust manipulation via taming distributional inconsistencies. *arXiv preprint arXiv:2602.09021*, 2026.

[84] Jiawen Yu, Hairuo Liu, Qiaojun Yu, Jieji Ren, Ce Hao, Haitong Ding, Guangyu Huang, Guofan Huang, Yan Song, Panpan Cai, et al. ForceVLA: Enhancing VLA models with a force-aware MoE for contact-rich manipulation. *NeurIPS*, 2025.

[85] Chengbo Yuan, Zicheng Zhang, Mingjie Zhou, Wendi Chen, Yi Wang, Zhuoyang Liu, Dantong Niu, Shuo Wang, Hui Zhang, Wenkang Zhang, et al. FTP-1: A generalist foundation tactile policy across tactile sensors for contact-rich manipulation. *arXiv preprint arXiv:2606.13102*, 2026.

[86] Wenzhen Yuan, Siyuan Dong, and Edward H. Adelson. GelSight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 2017.

[87] Yujie Zang, Yuhang Zheng, Xian Nie, Yupeng Zheng, Shuai Tian, Songen Gu, Chen Gao, Zining Wang, Shuicheng Yan, and Wenchao Ding. TacForeSight: Force-guided tactile world model for contact-rich manipulation. *arXiv preprint arXiv:2606.11184*, 2026.

[88] Chi Zhang, Penglin Cai, Ziheng Xi, Haoqi Yuan, Hao Luo, Wanpeng Zhang, Sipeng Zheng, Chaoyi Xu, and Zongqing Lu. Human-centric transferable tactile pre-training for dexterous robotic manipulation. *arXiv preprint arXiv:2607.01067*, 2026.

[89] Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. VLABench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. *ICCV*, 2025.

[90] Zongzheng Zhang, Haobo Xu, Zhuo Yang, Chenghao Yue, Zehao Lin, Huan-ang Gao, Ziwei Wang, and Hao Zhao. TA-VLA: Elucidating the design space of torque-aware vision-language-action models. *CoRL*, 2025.

[91] Zhemeng Zhang, Jiahua Ma, Xincheng Yang, Xin Wen, Yuzhi Zhang, Boyan Li, Yiran Qin, Jin Liu, Can Zhao, Li Kang, et al. TouchGuide: Inference-time steering of visuomotor policies via touch guidance. *RSS*, 2026.

[92] Zhiyuan Zhang, Pokuang Zhou, Kaidi Zhang, Adeesh Desai, Temitope Amosa, Davood Soleymanzadeh, Jiuzhou Lei, Minghui Zheng, and Yu She. ContactWorld: What matters in vision-tactile world models for contact-rich manipulation. *arXiv preprint arXiv:2606.13877*, 2026.

[93] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *RSS*, 2023.

[94] Yuhang Zheng, Songen Gu, Weize Li, Yupeng Zheng, Yujie Zang, Shuai Tian, Xiang Li, Ce Hao, Chen Gao, Si Liu, et al. OmniVTA: Visuo-tactile world modeling for contact-rich robotic manipulation. *arXiv preprint arXiv:2603.19201*, 2026.

[95] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. *CVPR*, 2019.

[96] Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *RSS*, 2025.

[97] Xinyue Zhu, Binghao Huang, and Yunzhu Li. Touch in the wild: Learning fine-grained manipulation with a portable visuo-tactile gripper. *NeurIPS*, 2025.

A Data Card

Table 3 summarizes NeoData, the released corpus, at the level of detail expected of a data card. Collection protocols, sensor specifications, and per-repository provenance are documented in full in the companion data technical report [51].

Table 3 Data card for NeoData, the released corpus.

Field	Value
Modalities	3 RGB views per episode (one third-person, two wrist-mounted), a language instruction, robot proprioceptive state, and 2 tactile streams per episode on single-arm platforms or 4 on dual-arm platforms, each from the self-developed visuo-tactile sensor (Section 3).
Format	LeRobot dataset format [13], holding per-view video files, tabular action/state records, and per-episode metadata.
Frame rate	30 fps, uniform across every modality, view, and platform.
Resolution	Raw capture up to 640×480 per RGB view. Tactile streams are captured at native sensor resolution rather than padded or upsampled at collection time. All streams are resized to 224×224 for model consumption.
Platforms	Single- and dual-arm configurations spanning the ARX X5, UR5e, Flexiv, Franka, and Piper arms and the Neo TacUMI handheld gripper (see the companion data report [51]).
Scale	Composition and episode accounting in the companion data report [51].
Action/state schema	Canonical 32-dimensional container. The first 20 dimensions hold one 10-dimensional slot per arm, each carrying a 3-dimensional position, a 6-dimensional rot6d rotation, and a 1-dimensional gripper channel, and the remaining 12 are always zero (Section 3).
Language annotations	Curated per-task English instructions, with several paraphrase variants per task assigned deterministically per episode.

B Compute Infrastructure

$\mathcal{N}_0$-VTLA is trained with multi-node distributed training at cluster scale on modern accelerators. Simulation post-training and smaller fine-tuning jobs run at correspondingly smaller scale on the same class of hardware. We describe the infrastructure at this level of generality and omit hardware models, device counts, memory footprints, and throughput figures.

C Data-Engineering Pitfalls

Section 3 defers to this appendix the individual invariants behind data quality verification. Every converted repository is checked against the invariants in Table 4 before it is allowed into training. Each pairs a red line with the symptom it produces when violated.

Table 4 Data-engineering invariants enforced before training and the symptom each produces when violated.

Red line	Symptom if violated
Actions stored as absolute end-effector poses on disk	Training loss converges normally, but the resulting policy is unusable on hardware.
A single coordinate frame and unit convention (meters) for state and action	The normalization sanity check on the mean position channel fails, well above its expected near-zero value.
Maximum group-of-pictures size of 30 frames per video	Training throughput collapses under periodic stalls that starve the trainer of data, since every random read decodes from the nearest keyframe.
Tactile frames stored at native sensor resolution, never padded or upsampled at collection	Wasted resolution, or, if padding is applied, spurious fixed content baked into every frame.
Every episode begins with a static, contact-free baseline period	The zero-contact reference frame used throughout the contact-difference tactile processing of Section 3 is no longer valid.
Corrupt video files are culled before entering training	A single corrupt file can silently kill a data-loading worker mid-run.
Any in-place data change is followed by purging the cached, pre-built training dataset	Training silently reads a stale cached dataset. The loss trajectory starts anomalously high and plateaus without converging.
Normalization statistics are computed under the same chunk-relative action transform used at training time	Training proceeds and the loss looks healthy, but the model learns at the wrong action scale.
Dataset output root matches the configured dataset root directory rather than a nested subdirectory	Repositories are written to a nested path and become undiscoverable by the training data loader.

D Deployment Protocol

Section 4 defers deployment-time details of the trained policy to this appendix, namely the execution contract for a predicted action chunk, the tactile baseline convention used at serve time, and the general shape of the serving interface.

Full action-chunk execution. At serve time the action expert emits an action chunk spanning the full training horizon of $H = 50$ steps set in Section 2.1, from one prefix encoding and one denoising solve of ten Euler steps per policy request. The deployment contract requires the controller to execute every step of a returned chunk before requesting the next one. This is not a convenience default. Executing only a truncated prefix of each chunk, fewer than the full 50 steps, before re-predicting can leave the arm at the target position with the gripper never closed, because the gripper-closing commands are concentrated in the tail of the chunk that early re-planning discards. This behavior is a property of the execution schedule, not of the underlying model, which closes the gripper correctly under full-chunk execution.

Tactile baseline reset. The tactile pathway of Section 2.2 conditions on a per-episode contact-difference baseline rather than on raw tactile frames. At serve time, a reset issued at the start of an episode clears any previously stored baseline. The first observation received after a reset is captured as the new zero-contact reference for every active tactile view, so the tactile difference at that instant is exactly zero, matching the convention established at collection time in Section 3. Every subsequent observation in the episode is differenced against that same stored baseline until the next reset. If a given view's tactile stream is unavailable, the controller falls back to treating that view as its own baseline, that is, no contact, rather than failing the request.

Serving interface. The trained policy is served behind a message-based request/response interface. A reset message clears the stored tactile baseline for a new episode. A predict request carries the current observation, comprising camera images, active tactile views, robot state, and an optional language instruction, and the interface returns the predicted action chunk together with basic timing information. We describe the interface at this level of generality and omit transport-level and deployment-script details.

E Per-Task Results and Scoring Rubrics

This appendix lists the point-based progress rubrics that define the progress score of Section 5.1 for the rubric-scored NeoReal tasks. Each rubric decomposes a task into a fixed sequence of subtask checkpoints with a point value each, so that the progress score is a reproducible sum of checkpoint credit rather than a subjective judgment.

Rubric-scored NeoReal tasks. Every NeoReal task is scored on a 100-point stage rubric. Table 5 lists the checkpoint sequences for two representative long-horizon tasks. The per-checkpoint point weights, which distribute the 100 points within each task, are documented in the companion data report [51].

Table 5 Checkpoint sequences for two representative long-horizon NeoReal task rubrics. Each task's stage rubric distributes 100 points across the checkpoints listed.

Task	Checkpoint sequence
Towel Folding	Take one towel from the pile Lift it Shake it flat Fold it into a square Place it aside
Bag Packing	Pull the bag close Open it Put items in Arrange them Place the bag aside

Representative rollouts. Figure 11, referenced from Section 5.2, complements the aggregate scores with keyframe strips of $\mathcal{N}_0$-VTLA executions on three NeoReal tasks and one NeoSim task.

F ALTER Offline Policy-Learning Details

Deployment-corpus composition. The screened deployment corpus contains the three mutually exclusive episode categories of Section 4.4. Towel Folding uses 351 clean demonstrations, 286 autonomous rollouts, and 180 HIL rollouts. Bag Packing uses 213, 243, and 238 episodes, respectively. Cardboard Box Folding uses 550, 587, and 397 episodes, respectively. The HIL totals include the separately collected staged-recovery episodes that begin from selected error states; these subsets contain 80, 72, and 110 episodes for the three tasks, respectively.

Successful-execution throughput. Figure 12 complements the main-text success-rate comparison with the measured number of successful task completions per hour under the evaluation protocol.

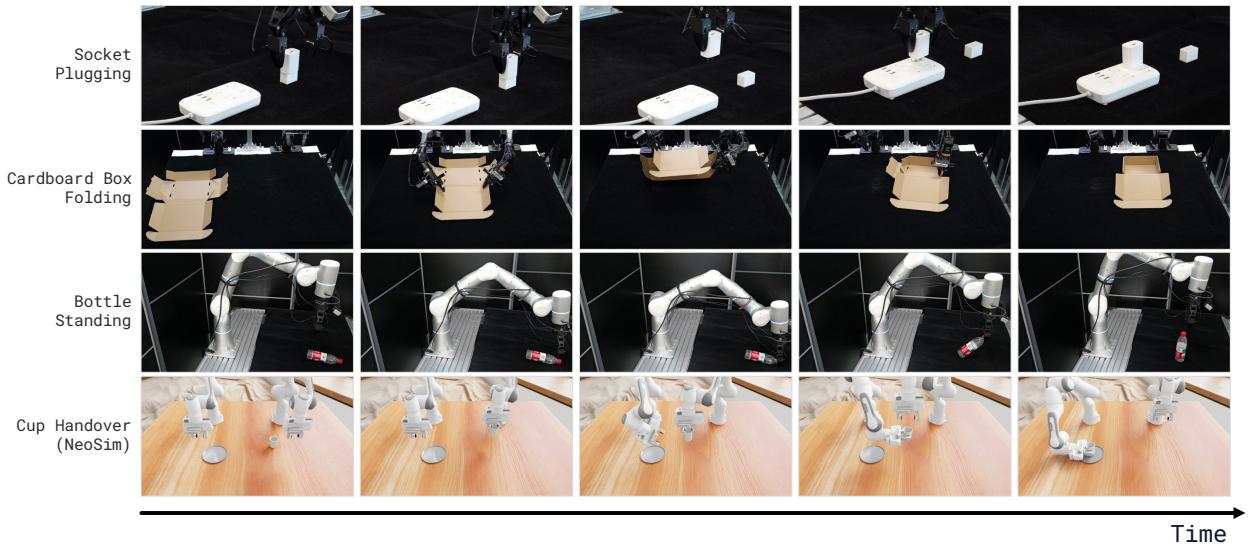

Figure 11 Representative rollouts. Keyframe strips of $\mathcal{N}_0$-VTLA executions over time: Socket Plugging, Cardboard Box Folding, and Bottle Standing from NeoReal, and the dual-arm Cup Handover task in NeoSim. Aggregate results are plotted in Figure 7.

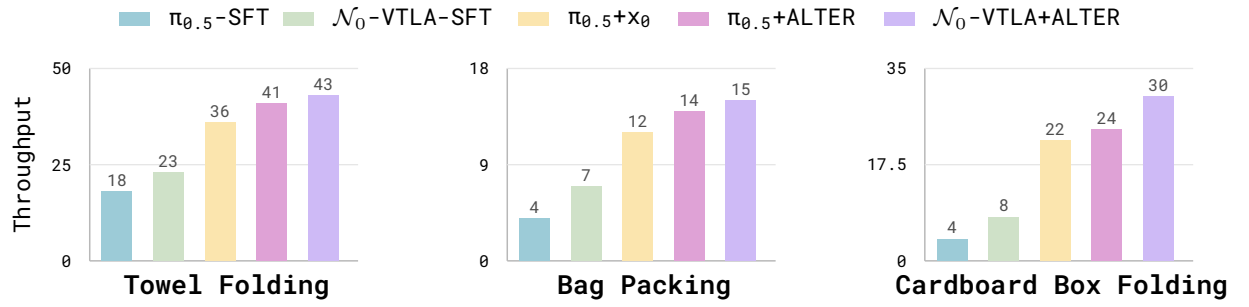

Figure 12 Successful-execution throughput under supervised and offline policy learning. All bars report measured successful task completions per hour. Vertical-axis ranges differ across tasks and are labeled separately.

Representative stage annotations. Figure 13 shows the task-specific stage granularity used to construct the clean-demonstration progress targets of Section 4.4. Towel Folding and Bag Packing use L1 stages. Cardboard Box Folding uses L2 stages because its L1 decomposition is overly fine-grained and its L2 stages already provide sufficient task structure.

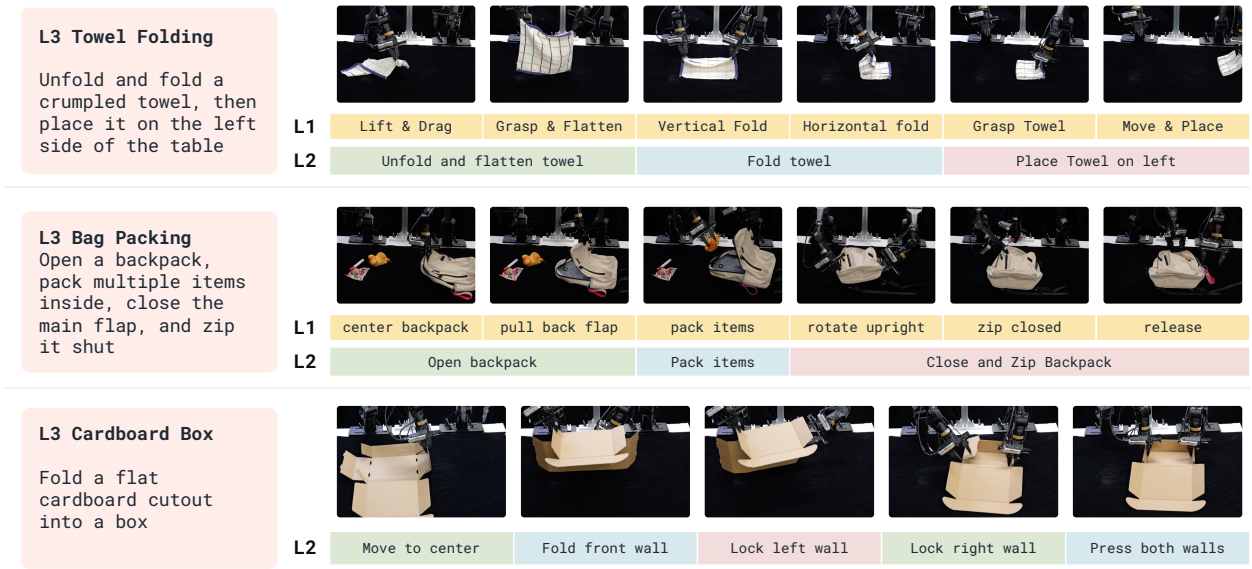

Figure 13 Representative stage annotations used for progress-model supervision. Towel Folding and Bag Packing use L1 action stages, with their parent L2 stages shown below. Cardboard Box Folding uses L2 stages.

Implementation details. For event supervision we use margin $m = 0.02$. In implementation, event pairs comprise approximately 5% of sampled pairs and receive a per-sample loss multiplier of 0.1. Before assigning stages, we apply a five-frame median filter to the global-progress predictions. We then enforce non-decreasing stage indices by replacing each predicted index with the maximum index observed up to that frame. These operations prevent short-term prediction noise from moving a sample back to an earlier stage. They do not modify the local-change estimate used for within-stage ranking. Because a staged-recovery episode begins from a selected error state, its initial global progress cannot be assumed to be zero. We estimate the offset δ_e in Equation 8 by comparing its first observation with the starts of three clean reference demonstrations. If fewer than H future frames remain, the local score is normalized by the ratio of the full action-chunk horizon to the available horizon. The terminal self-comparison is zero. During policy training, the advantage tag is omitted with probability 0.3, so the policy also sees the original task prompt without the additional text input.

Progress trajectories on additional tasks. Figure 14 shows our pairwise progress model on the two tasks omitted from the main-text diagnostic. In both cases, predicted progress decreases when execution degrades and rises again after the robot returns to a productive execution state.

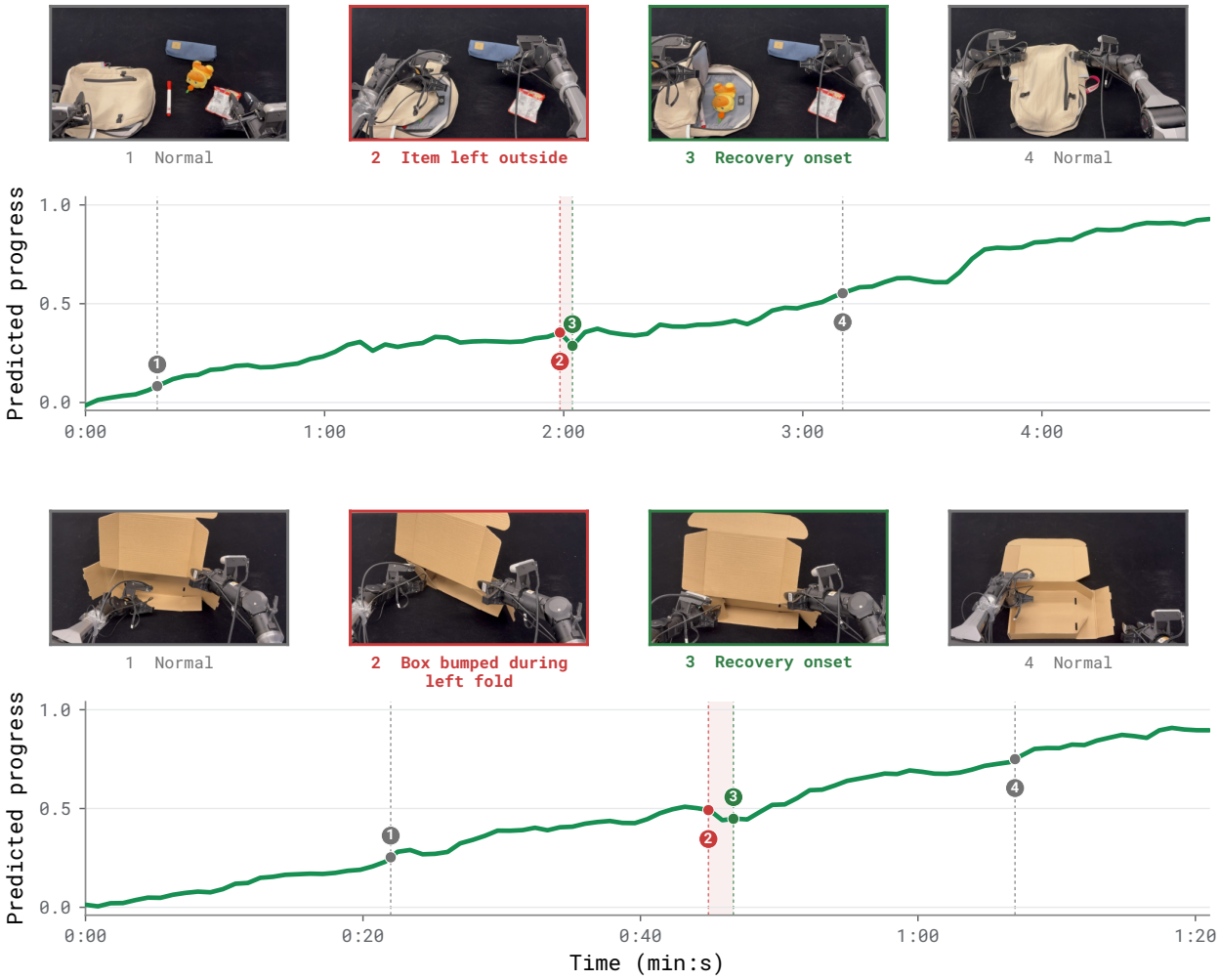

Figure 14 Predicted progress on additional deployment tasks. One evaluation episode per task, held out from progress-model training and scored by our pairwise progress model. The curve is the predicted global progress $\hat{\phi}_t$ against the episode's first frame. Numbered curve markers correspond to the synchronized frames above each panel. Gray denotes normal execution, red the onset of degradation, and green the onset of recovery. The pale red interval spans the detected degradation. In the Bag Packing episode, the policy starts closing the bag while an item remains on the table before recovering. In Cardboard Box Folding, the arm accidentally knocks the box out of alignment while folding the left side, and the subsequent correction restores its pose.