# FLEXSENSE: FLEXIBLE SENSOR CONFIGURATIONS WITH A UNIFIED VISION-LANGUAGE-ACTION MODEL

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Vision-Language-Action (VLA) models increasingly generalize across tasks, environments, and robot embodiments, yet most remain coupled to a fixed sensory interface. In practice, robots expose different combinations of optional sensors, and training a separate policy for every configuration scales poorly and prevents demonstrations collected under different sensing setups from being fully shared. We introduce *FlexSense*, a unified VLA framework for robotic manipulation under heterogeneous sensor configurations. It consists of three modules: First, *Shared-Residual Sensory Encoding (ShareRes)* captures transferable control structure through a shared sensory backbone while preserving sensor-specific cues through lightweight residual adapters gated by zero-initialized scalars. Second, *Sensor-Set-Conditioned FiLM (SetFiLM)* explicitly conditions each action block on the active sensor set, complementing token masking with configuration-aware action generation. Third, *Utility-Guided Sensor Selection (UtilSelect)* derives subset-dependent supervision from counterfactual action errors of the frozen policy, distills it into a lightweight scorer, and selects the sensor subset with the lowest predicted action error at each action-chunk query. We evaluate FlexSense on LIBERO, ManiSkill, and real-world robotic manipulation across multiple sensor configurations. Across LIBERO and ManiSkill, FlexSense remains within **1.6 percentage points** of the corresponding configuration-specific specialist for every tested sensor configuration. Real-robot experiments further show that the same policy benefits from complementary depth and tactile sensing, reaching 100% success on Pick-and-Place and 80% on Precision Plug Insertion. Our code will be released.

## 1 INTRODUCTION

Vision-Language-Action (VLA) models are becoming increasingly general across manipulation tasks, environments, and robot embodiments (Zitkovich et al., 2023; Kim et al., 2025; Black et al., 2025b;a; Intelligence et al., 2026; Open X-Embodiment Collaboration et al., 2025; NVIDIA et al., 2025; Doshi et al., 2025; Octo Model Team et al., 2024). Their sensory interfaces, however, remain considerably less general. Most VLA policies assume that the same observation channels, encoders, and fusion pathways are available during both training and deployment. Real robots violate this assumption: one platform may provide only RGB observations and proprioception, whereas another may additionally include depth, force/torque, tactile, event-based, or other physical sensors (Fang et al., 2024; He et al., 2025; Chen et al., 2023; Yu et al., 2025; Qu et al., 2025; Zhai et al., 2026). Sensor availability may also change across hardware generations, maintenance cycles, or deployment settings (Skand et al., 2025; Piekenbrinck et al., 2024; Wei et al., 2025a). With $n$ optional sensors, as many as $2^n$ configurations are possible, making it impractical to train and maintain a separate policy for every configuration (Zhang et al., 2026). This motivates a configuration-compatible VLA: one policy that can use demonstrations collected under heterogeneous sensing setups and operate under different supported sensor combinations without configuration-specific retraining.

Achieving this goal requires more than masking unavailable sensor tokens. It presents three coupled challenges. **First, cross-configuration representation sharing.** Different sensors have heterogeneous raw structures but often encode common task and control information. Fully independent encoders preserve sensor-specific cues but prevent knowledge transfer, whereas a fully shared encoder may suppress information unique to depth, force, or tactile observations (Zhu et al., 2024;



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Jones et al., 2025; Wang et al., 2023; **?**). **Second, configuration-aware action generation.** Removing unavailable tokens controls what information the policy can access, but it does not explicitly inform the shared action generator which evidence is currently available. The same sensory feature can have different implications depending on the other sensors present, so the action computation itself should be conditioned on the active sensor set. **Third, set-dependent sensor utilization.** Sensor value is neither static nor independent: depth may be most useful during geometric alignment, force or tactile feedback may become critical after contact, and a sensor that is useful alone may become redundant when another sensor supplies overlapping evidence (Yang et al., 2024; Liu et al., 2025; Lan et al., 2025; Feng et al., 2025). A unified policy must therefore share what is transferable, condition its computation on what is available, and activate what is useful for the current observation.

We address these challenges with *FlexSense*, a unified VLA framework that treats sensor configuration as explicit policy context. FlexSense distinguishes the physically available sensor set $C$ from the subset $S_t \subseteq C$ activated for the current action chunk. It extends a pretrained VLA while leaving its native RGB, language, and proprioceptive pathways unchanged. **First**, to enable cross-configuration transfer without erasing sensor-specific information, *Shared-Residual Sensory Encoding (ShareRes)* maps each optional sensor through a modality-specific input stem, a sensory backbone shared across modalities, and a lightweight modality-specific residual adapter. The shared backbone captures reusable control structure, while zero-initialized scalar gates on the residual branches and residual regularization preserve complementary sensor-specific cues without dominating the common pathway. Configuration-constrained modality dropout further exposes the policy to feasible sensor subsets while never synthesizing a modality absent from the original demonstration. **Second**, to make action generation configuration aware, *Sensor-Set-Conditioned FiLM (SetFiLM)* encodes the active sensor set and injects zero-initialized scale and shift terms into every action block (Perez et al., 2018). The availability mask determines which sensor tokens are visible, whereas SetFiLM determines how the shared action generator processes the resulting configuration. **Third**, to support observation-dependent sensor use, *Utility-Guided Sensor Selection (UtilSelect)* evaluates candidate subsets with a frozen policy and measures their expert-referenced action errors using shared flow noise. These subset-conditioned errors define action-grounded marginal utilities for interpretation and directly supervise a lightweight set-error scorer. At deployment, the scorer selects the subset with the lowest predicted action error at every action-chunk query, allowing sensor use to vary over the course of a manipulation trajectory without changing policy parameters.

We evaluate *FlexSense* on LIBERO (Liu et al., 2023), ManiSkill (Tao et al., 2025), and an ARX-X5 real-robot platform. The experiments are organized to test both end-task performance and the mechanisms underlying configuration compatibility. On the standard LIBERO benchmark, *FlexSense* achieves a 97.4% average success rate, remaining competitive with strong reported baselines. More importantly, a single checkpoint evaluated under no auxiliary sensing, Depth, Force, and Depth+Force obtains a configuration average of 97.14%, slightly exceeding the 96.80% average of four separately trained specialists. On five ManiSkill tasks, *FlexSense* achieves a 59.2% average success rate, compared with 47.4% for the listed $\pi_{0.5}$ baseline. The real-robot experiments evaluate one policy under different combinations of depth and tactile sensing. Combined sensing raises success to 100% on Pick-and-Place and 80% on Precision Plug Insertion, compared with 85% and 20%, respectively, without optional sensing.

Our main contributions are as follows.

- We propose *FlexSense* and formulate configuration-compatible VLA learning, enabling one policy to operate across heterogeneous sensor sets without imputing unavailable modalities or configuration-specific retraining.
- We develop *ShareRes* and *SetFiLM* to share transferable control knowledge, preserve sensor-specific cues, and condition action generation on the active sensor set.
- We introduce *UtilSelect*, which learns set-dependent utility from counterfactual action errors and adaptively selects sensors at each action-chunk query.

## 2 RELATED WORKS

**Vision-Language-Action Models.** Vision-Language-Action (VLA) models leverage pretrained vision-language representations and large-scale robot demonstrations to learn generalist manipulation policies (Black et al., 2025b; Zitkovich et al., 2023; Kim et al., 2025). Existing approaches generate actions either through autoregressive action representations (Zitkovich et al., 2023; Kim et al., 2025; Pertsch et al., 2025) or through continuous diffusion-based action experts Black et al. (2025b;a); NVIDIA et al. (2025); Wen et al. (2025); Li et al. (2024). Beyond advances in action generation, recent work has enriched the perceptual interface of robot policies with additional physical sensing (Liu et al., 2026; Zhai et al., 2026; Wei et al., 2025b; Qu et al., 2025; Yu et al., 2025). Geometry-aware policies leverage depth, point clouds, or explicit spatial representations to improve spatial reasoning (Li et al., 2026; Qu et al., 2025; Yuan et al., 2025), while tactile, force, and auditory signals provide local interaction cues that are particularly valuable under visual occlusion and during contact-rich control (Li et al., 2023; Yu et al., 2025; Huang et al., 2025; Zhang et al., 2025). These works demonstrate the benefits of enriching robot policies with additional physical sensing.

**Robot Policies under Heterogeneous Sensor Configurations.** Recent robot learning studies have explored policies that operate with heterogeneous sensory inputs and varying observation spaces. Earlier efforts considered adaptation to new observation spaces, cross-embodiment learning, and unified modeling of multimodal inputs (Octo Model Team et al., 2024; Doshi et al., 2025; Zhu et al., 2024). More recent work has extended these ideas to generalist and VLA-based policies, with increasing attention to heterogeneous physical sensing, adaptive sensor invocation, and robustness to missing auxiliary sensors (Jones et al., 2025; Hirose et al., 2025; Liu et al., 2026; Zhang et al., 2026). Related multimodal learning has developed complementary mechanisms for varying or incomplete modality sets, including missing-modality training, shared–specific representations, prompting, and flexible routing (Girdhar et al., 2023; 2022; Ma et al., 2022; Wang et al., 2023; Lee et al., 2023; Yun et al., 2024). Together, these works establish important foundations for flexible policy learning under heterogeneous sensory inputs. Our work complements these directions by focusing on sensor-configuration generalization within a unified VLA.

## 3 METHOD

### 3.1 OVERVIEW OF FLEXSENSE

**Problem Formulation.** We consider robotic manipulation with a set of optional physical sensor modalities $\mathcal{M}$, in addition to the native RGB, language, and proprioceptive inputs of the underlying VLA. A sensor configuration $\mathcal{C} \subseteq \mathcal{M}$ specifies which optional modalities are available. Given observations under configuration $\mathcal{C}$, FlexSense predicts an $H$-step action chunk $\mathbf{A}_t = [\mathbf{a}_t, \dots, \mathbf{a}_{t+H-1}]$ using a single configuration-compatible policy, $\mathbf{A}_t \sim \pi_\theta(\cdot \mid \mathbf{o}_t^{\mathcal{C}}, \mathcal{C})$. The same policy parameters are shared across all supported sensor configurations.

The training corpus may contain demonstrations collected under native configurations $\mathcal{C}_i$. A modality absent from $\mathcal{C}_i$ is not treated as a corrupted observation to be reconstructed; it was not observed for that demonstration. FlexSense therefore aims to transfer common task and control structure across heterogeneous sensing setups while preserving sensor-specific information.

**Framework Overview.** As shown in Fig. 1, FlexSense extends a pretrained VLA with three lightweight components. ShareRes transfers knowledge across sensor configurations while retaining sensor-specific cues; SetFiLM adapts the action policy to the active sensor set; and UtilSelect selects an observation-dependent subset $\mathcal{S}_t \subseteq \mathcal{C}$ for action generation. Here, $\mathcal{C}$ denotes the sensors available to the policy, whereas $\mathcal{S}_t$ denotes those actively used for the current action chunk.

### 3.2 SHARERES: SHARED-RESIDUAL SENSORY ENCODING

ShareRes leaves the native RGB, language, and proprioceptive pathways unchanged. Each optional sensor $m \in \mathcal{M}$ uses a lightweight modality-specific stem $E_m$ to map its raw observation $\mathbf{x}_t^m$ to tokens $\mathbf{T}_t^m \in \mathbb{R}^{N_m \times d_s}$, using CNN-based stems for image-like inputs and MLP-based stems for low-dimensional physical signals. To share transferable control structure without suppressing modality-specific cues, ShareRes combines a sensory backbone shared across optional modalities

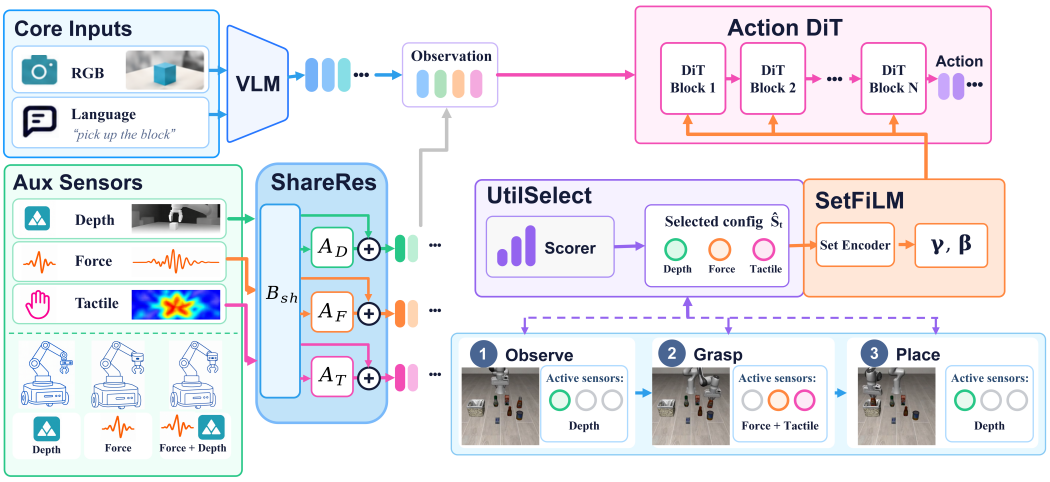


Figure 1: **Overview of FlexSense** (§3). FlexSense augments a pretrained VLA with ShareRes, SetFiLM, and UtilSelect, enabling one policy to share sensory knowledge across configurations, adapt action generation to the active sensor subset, and select useful sensory inputs online.

with lightweight modality-specific residual adapters (**?**Wang et al., 2023):

$$\begin{aligned} \mathbf{S}_t^m &= B_{\text{sh}}(\text{LN}_m(\mathbf{T}_t^m)), \mathbf{R}_t^m = A_m(\text{LN}_m(\mathbf{T}_t^m)), \\ \mathbf{\Delta}_t^m &= \gamma_m \mathbf{R}_t^m, \qquad \mathbf{Z}_t^m = P_m(\mathbf{S}_t^m + \mathbf{\Delta}_t^m) + \mathbf{e}_m. \end{aligned} \tag{1}$$

Here, $B_{\text{sh}}$ is a two-layer Transformer shared across optional modalities, while $A_m$ is a modality-specific bottleneck adapter. $P_m$ projects to the action-policy width and $\mathbf{e}_m$ identifies the sensor type. The learnable scalar $\gamma_m$ is initialized to zero, so each modality-specific branch begins as a residual correction to the shared representation.

This decomposition separates two competing requirements. Fully independent encoders preserve sensor-specific structure but limit transfer across configurations, whereas a completely shared encoder may suppress cues unique to individual sensors. ShareRes instead lets $B_{\text{sh}}$ absorb transferable structure while $A_m$ provides a lightweight sensor-specific correction.

To prevent the specific branches from dominating the shared pathway, we regularize the magnitude of the active residual corrections:

$$\mathcal{L}_{\text{res}} = \sum_{m \in \mathcal{M}} \frac{\sum_b a_{b,m} \,\text{mean}\left[(\mathbf{\Delta}_b^m)^2\right]}{\sum_b a_{b,m} + \epsilon}, \qquad \mathcal{L} = \mathcal{L}_{\text{act}} + 10^{-4}\mathcal{L}_{\text{res}}, \tag{2}$$

where $a_{b,m} = \mathbf{1}[m \in \widetilde{\mathcal{C}}_b]$ indicates whether modality $m$ remains active for sample $b$ after configuration-constrained modality dropout. In our flow-matching VLA instantiations (Black et al., 2025b;a; Community, 2026), $\mathcal{L}_{\text{act}} = \mathcal{L}_{\text{flow}}$. This regularization encourages transferable structure to remain in the shared pathway while allowing modality-specific corrections.

### 3.3 SetFiLM: Sensor-Set-Conditioned FiLM

Let $\mathbf{Z}_{t,l}^{\text{base}}$ denote the conditioning representation supplied by the native VLA pathway to action block $l$. FlexSense augments it with optional sensory tokens, yielding $\mathbf{M}_{t,l} = [\mathbf{Z}_{t,l}^{\text{base}}; \{\mathbf{Z}_t^m\}_{m \in \mathcal{M}}]$, while a configuration-dependent attention mask removes unavailable modalities. Masking controls which sensory information is visible, but does not explicitly condition the shared action computation on the sensor-set composition.

SetFiLM therefore encodes the active sensor subset with a binary vector $\mathbf{c}_t$, where $\mathbf{c}_t[m] = \mathbf{1}[m \in \mathcal{S}_t]$, and injects it into each action block through FiLM-style scale and shift modulation (Perez et al., 2018):

$$\widetilde{\mathbf{H}}_{t,l} = \text{LN}(\mathbf{H}_{t,l}) \odot [1 + \mathbf{s}_l(\tau) + \boldsymbol{\gamma}_l(\mathbf{c}_t)] + \mathbf{b}_l(\tau) + \boldsymbol{\beta}_l(\mathbf{c}_t), \tag{3}$$

where $\mathbf{s}_l(\tau)$ and $\mathbf{b}_l(\tau)$ are inherited from the base action expert. The added configuration-conditioned projections are zero initialized, making SetFiLM a no-op at initialization. Thus, masking determines *what* sensory information is available, while SetFiLM determines *how* the shared action generator processes that configuration.

The configuration supplied to SetFiLM always matches the subset exposed by the attention mask. During training it corresponds to the sampled feasible sensor subset, while at deployment it corresponds to the subset selected for the current action chunk. SetFiLM therefore conditions on the evidence actually used by the policy rather than merely the sensors installed on the robot.

**Training across sensor configurations.** To avoid over-dependence on the full sensor set, we apply stochastic modality dropout during unified policy training (Neverova et al., 2016; Skand et al., 2025; Zhang et al., 2026). For a demonstration with native configuration $\mathcal{C}_i$, optional modalities are dropped only from those originally present, producing a feasible subset $\widetilde{\mathcal{C}}_i \subseteq \mathcal{C}_i$. Modalities absent from the original demonstration are never synthesized or imputed. The same sampled subset is used by both the attention mask and SetFiLM, and all configurations optimize the native action objective together with Eq. 2.

### 3.4 UTILSELECT: UTILITY-GUIDED SENSOR SELECTION

The value of an auxiliary sensor varies with the available sensor set and the current manipulation stage (Feng et al., 2025). We therefore derive observation-level sensor utility from the action errors of the policy rather than assigning a fixed task-level importance score (Hu et al., 2022).

**Action-grounded utility supervision.** After training the unified policy, we freeze it and evaluate candidate subsets $S \subseteq \mathcal{C}$ against the expert action chunk. All subsets within a repeat use the same initial flow noise to reduce sampling-induced variation. Let $\widehat{\mathbf{A}}_{t,k}^{S}$ denote the prediction under subset $S$ in repeat $k$. We define

$$\begin{aligned} E_t(S) &= \frac{1}{K}\sum_{k=1}^{K} \mathrm{SmoothL1}\left(\widehat{\mathbf{A}}_{t,k}^{S}, \mathbf{A}_t^{\star}\right), \\ u_{m,t}^{\star}(S) &= \frac{E_t(S) - E_t(S\cup\{m\})}{E_t(S) + E_t(S\cup\{m\}) + \epsilon}, \qquad m \in \mathcal{C}\setminus S. \end{aligned} \tag{4}$$

SmoothL1 is averaged over all action dimensions and time steps in the predicted action chunk. $E_t(S)$ measures the frozen policy's imitation error under subset $S$, while $u_{m,t}^{\star}(S)$ measures the marginal reduction in action error obtained by adding sensor $m$. The utility is therefore subset dependent and should not be interpreted as causal sensor importance. The marginal utility characterizes the contribution of adding a sensor, whereas the scorer below is trained on set-level errors because online selection directly requires ranking candidate subsets.

Using set-level action error rather than an independent score for each sensor preserves interactions between modalities: the contribution of one sensor can change when another sensor is already present. This allows UtilSelect to model complementarity and redundancy between sensors instead of assuming that their values are additive.

**UtilSelect scorer and online selection.** Because computing $E_t(S)$ requires expert actions and repeated policy evaluations, it is used only for offline supervision. We distill these errors into a lightweight scorer $g_\phi$. Frozen RGB/vision-language, proprioceptive, and optional-sensor representations are pooled into source-level summaries with source-type embeddings, while a candidate-subset embedding queries them through a small cross-attention module. An MLP with a Softplus output predicts $\widehat{E}_t(S) = g_\phi(\mathbf{o}_t^{\mathcal{C}}, \mathcal{C}, S) \geq 0$ and is trained with SmoothL1 regression while the VLA remains frozen. At each action-policy query, the scorer evaluates feasible subsets and selects

$$\widehat{\mathcal{S}}_t = \arg\min_{S\subseteq\mathcal{C}} \widehat{E}_t(S). \tag{5}$$

The selected subset determines both the sensory mask and sensor-set conditioning used by the same action policy, which then generates $\widehat{\mathbf{A}}_t = \pi_\theta(\cdot \mid \mathbf{o}_t^{\widehat{\mathcal{S}}_t}, \widehat{\mathcal{S}}_t)$. Selection is recomputed at every action-chunk query, allowing sensor use to vary over the trajectory. For the small sensor sets considered in our experiments, all feasible subsets are scored directly.

Table 1: **Comparison with existing policies** on LIBERO (Liu et al., 2023) (§4.2.1). All numbers are task success rates (%).

| Method | Spatial | Object | Goal | Long | Average |
|---|---|---|---|---|---|
| OpenVLA [CoRL24] | 84.7 | 88.4 | 79.2 | 53.7 | 76.5 |
| OpenVLA-OFT [RSS25] | 97.6 | 98.4 | 97.9 | 94.5 | 97.1 |
| $\pi_0$ [RSS25] | 96.8 | 98.8 | 95.8 | 85.2 | 94.2 |
| $\pi_0$+FAST [RSS25] | 96.4 | 96.8 | 88.6 | 60.2 | 85.5 |
| $\pi_{0.5}$ [CoRL25] | **98.8** | 98.2 | 98.0 | 92.4 | 96.9 |
| GR00T-N1.5 [arXiv25] | 92.0 | 92.0 | 86.0 | 76.0 | 86.5 |
| StarVLA-$\pi$ [arXiv26] | **98.8** | **99.6** | 95.8 | 88.4 | 95.7 |
| **FlexSense (Ours)** | 97.0 | 99.2 | **98.4** | **95.0** | **97.4** |

# 4 EXPERIMENTS

We evaluate FlexSense on LIBERO (Liu et al., 2023), ManiSkill (Tao et al., 2025), and real-world manipulation. Our experiments examine whether configuration compatibility preserves standard task performance, whether one policy can replace configuration-specific specialists, whether heterogeneous demonstrations transfer across sensor configurations, and whether the proposed modules enable configuration-aware and observation-dependent sensor use.

## 4.1 EXPERIMENTAL SETUP

**Simulation Setup.** RGB, language, and proprioception are always available, while Depth ($D$) and Force ($F$) are optional, yielding $\varnothing$, $D$, $F$, and $D + F$. Depth is obtained from simulator rendering, while Force is represented as a 6-D wrist wrench $\mathbf{w}_t = [F_x, F_y, F_z, \tau_x, \tau_y, \tau_z]$.

**LIBERO.** We evaluate the Spatial, Object, Goal, and Long suites, each containing 10 tasks with 50 demonstrations per task. Starting from the StarVLA-$\pi$ LIBERO-4in1 checkpoint, we jointly fine-tune one policy across all four suites and evaluate each task over 50 rollouts.

**ManiSkill.** We evaluate PegInsertionSide, PickCube, StackCube, PlugCharger, and PushCube. Starting from $\pi_{0.5}$, we jointly train one policy on 1,000 successful official motion-planning demonstrations per task and evaluate each task over 200 episodes.

**Real-robot Setup.** We use an ARX-X5 (2023) robot equipped with two RealSense D436 RGB-D cameras, one fixed in a third-person position and one wrist-mounted, together with two Xense Photon fingertip tactile sensors. FlexSense is instantiated on $\pi_{0.5}$ and evaluated on Pick-and-Place and Precision Plug Insertion. RGB, language, and proprioception are available, while Depth ($D$) and Tactile ($T$) are optional, yielding $\varnothing$, $D$, $T$, and $D + T$. Demonstrations are collected through VR-controller teleoperation. The $\pi_{0.5}$ baseline is fine-tuned on the same 70 demonstrations per task and evaluated on the same 20 initial-state trials and evaluation horizon as FlexSense, but receives no optional depth or tactile input.

## 4.2 BENCHMARK AND CONFIGURATION PERFORMANCE

### 4.2.1 COMPARISON WITH EXISTING POLICIES

We first ask whether supporting heterogeneous sensor configurations compromises performance when the full $D + F$ configuration is available. Reported results from prior work are included as benchmark references; sensor inputs and training or evaluation protocols may differ, so they are not treated as controlled ablations.

**LIBERO.** Among the reported reference results, FlexSense achieves a $97.4\%$ average success rate, compared with $97.1\%$ for OpenVLA-OFT and $96.9\%$ for $\pi_{0.5}$, and obtains the strongest reported results on Goal and Long in Table 1. These results indicate that configuration compatibility can preserve strong standard manipulation performance under the full sensing setup.

**ManiSkill.** As shown in Table 2, FlexSense reaches $59.2\%$ average success, compared with $47.4\%$ for the listed $\pi_{0.5}$ baseline, while obtaining $95.0\%$ on PickCube and $12.0\%$ on PlugCharger. Its overall result is also close to the reported $61.0\%$ of Rodrigues-DP. Together with LIBERO, these

Table 2: **Comparison with existing policies** on ManiSkill(Tao et al., 2025) (§4.2.1). All numbers are task success rates (%).

| Method | PegInsertionSide | PickCube | StackCube | PlugCharger | PushCube | Average |
|---|---|---|---|---|---|---|
| RDT-1B [ICLR25] | 13.2 | 77.2 | 74.0 | 1.2 | **100** | 53.6 |
| Rodrigues-DP [ICLR26] | **58** | 94 | 44 | 10 | **100** | **61** |
| ACT [RSS23] | – | 20 | 30 | – | 76 | – |
| Diffusion Policy [RSS23] | 0 | 40 | **80** | 0 | 88 | 41.6 |
| OpenVLA [CoRL24] | 0 | 6 | 3 | 0 | 27 | 7.2 |
| Octo [RSS24] | 0 | 9 | 1 | 0 | 67 | 15.4 |
| SmolVLA [arXiv25] | – | – | 12.7 | – | 86.3 | – |
| $\pi_{0.5}$ [CoRL25] | 2.5 | 70.5 | 67.0 | 1.0 | 96.0 | 47.4 |
| **FlexSense (Ours)** | 14.0 | **95.0** | 75.0 | **12.0** | **100** | 59.2 |

Table 3: **Specialist-versus-unified comparison** on LIBERO (Liu et al., 2023) (§4.2.2). $D$ denotes Depth, $F$ denotes Force, and $\varnothing$ denotes no auxiliary sensor.

| Policy | Sensors | Spatial | Object | Goal | Long | Average |
|---|---|---|---|---|---|---|
| $\varnothing$ specialist | $\varnothing$ | 98.60 | 97.20 | 97.60 | 89.60 | 95.75 |
| $D$ specialist | $D$ | 98.60 | 98.40 | 97.00 | 94.40 | 97.10 |
| $F$ specialist | $F$ | 98.40 | 97.80 | 97.60 | 94.40 | 97.05 |
| $D + F$ specialist | $D + F$ | 98.40 | 98.60 | 98.00 | 94.20 | 97.30 |
| FlexSense | $\varnothing$ | 98.20 | 98.60 | 97.80 | 94.20 | 97.20 |
| FlexSense | $D$ | 98.20 | 98.40 | 97.20 | 94.00 | 96.95 |
| FlexSense | $F$ | 98.20 | 98.00 | 98.00 | 93.80 | 97.00 |
| FlexSense | $D + F$ | 97.00 | 99.20 | 98.40 | 95.00 | 97.40 |

reference comparisons indicate that configuration compatibility can retain strong performance under the full sensing setup.

### 4.2.2 Configuration-Compatible Learning

We test the configuration-compatibility claim: whether a single policy can replace policies trained separately for each sensor setup without introducing substantial negative interference.

**LIBERO.** As shown in Table 3, a single FlexSense checkpoint averages $97.14\%$ across configurations, compared with $96.80\%$ for four separately trained specialists. Importantly, parameter sharing introduces no systematic penalty: the gaps under $D$ and $F$ are only $-0.15$ and $-0.05$ points, while FlexSense exceeds the specialists under $\varnothing$ and $D + F$. This indicates that heterogeneous sensor configurations need not be isolated into separate policies; joint training can preserve configuration-specific behavior while sharing a common action policy.

Table 4: **Specialist-versus-unified comparison** on ManiSkill (Tao et al., 2025) (§4.2.2). $D$ denotes Depth, $F$ denotes Force, and $\varnothing$ denotes no auxiliary sensor.

| Policy | Sensors | PegInsertionSide | PickCube | StackCube | PlugCharger | PushCube | Average |
|---|---|---|---|---|---|---|---|
| $\varnothing$ specialist | $\varnothing$ | 7.5 | 86.0 | 67.0 | 2.5 | 100 | 52.6 |
| $D$ specialist | $D$ | 11.0 | 95.0 | 75.0 | 6.0 | 100 | 57.4 |
| $F$ specialist | $F$ | 15.5 | 90.5 | 72.5 | 12.5 | 100 | 58.2 |
| $D + F$ specialist | $D + F$ | 16.0 | 96.0 | 76.5 | 13.5 | 100 | 60.4 |
| FlexSense | $\varnothing$ | 8.5 | 87.0 | 68.0 | 3.0 | 99.0 | 53.1 |
| FlexSense | $D$ | 10.5 | 93.5 | 73.5 | 5.0 | 100 | 56.5 |
| FlexSense | $F$ | 13.5 | 90.0 | 70.5 | 10.5 | 100 | 56.9 |
| FlexSense | $D + F$ | 14.0 | 95.0 | 75.0 | 12.0 | 100 | 59.2 |

**ManiSkill.** Table 4 shows that the same behavior persists in the contact- and precision-sensitive ManiSkill setting. FlexSense reaches $56.4\%$ on average across configurations versus $57.2\%$ for four specialists, with a maximum absolute gap of $1.3$ points. Moreover, different sensors retain task-dependent benefits: Depth helps more on PickCube and StackCube, whereas Force contributes more on PegInsertionSide and PlugCharger. The small specialist gap despite these sensing requirements suggests that a shared policy need not collapse toward a single sensor-use pattern.

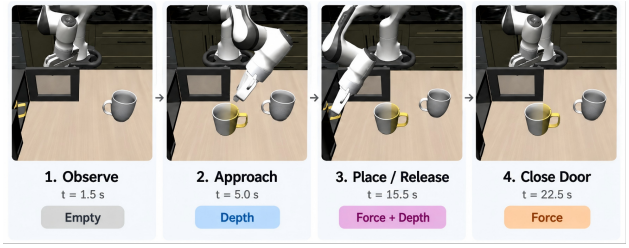


Figure 2: **Stage-wise sensor configurations** selected by UtilSelect along a successful LIBERO-10 rollout (§4.3).

Table 5: **Cross-configuration data utilization** on LIBERO-10 (Liu et al., 2023)(§4.3).

| Architecture | $D$ data coverage | Successes | Rate (%) |
|---|---|---|---|
| Independent | 10% | 445 | 89.0 |
| Independent | 50% | 447 | 89.4 |
| Independent | 100% | 449 | 89.8 |
| ShareRes | 10% | 446 | 89.2 |
| ShareRes | 50% | 457 | 91.4 |
| ShareRes | 100% | 468 | 93.6 |

Table 6: **Component ablation** on LIBERO (Liu et al., 2023) (§4.4). Each entry is the average success rate (%) over Spatial, Object, Goal, and Long.

| Model variant | $\varnothing$ | $D$ | $F$ | $D + F$ | Avg. |
|---|---|---|---|---|---|
| Base | 94.2 | 94.8 | 94.6 | 95.4 | 94.75 |
| Dropout only | 96.1 | 95.3 | 95.2 | 95.8 | 95.60 |
| Dropout + ShareRes | 96.2 | 95.9 | 95.8 | 96.3 | 96.05 |
| Dropout + SetFiLM | 96.5 | 95.8 | 95.7 | 96.4 | 96.10 |
| Dropout + ShareRes + SetFiLM | **97.2** | **96.2** | **96.4** | **96.6** | **96.60** |

### 4.3 Analysis of Representation Sharing and Sensor Utility

We next examine the mechanisms underlying configuration compatibility: whether demonstrations transfer across heterogeneous sensor configurations, and whether sensor use adapts within a manipulation trajectory.

**Representation Sharing.** Table 5 holds the $F$ training pool and evaluation configuration fixed while increasing additional $D$-only data from $10\%$ to $100\%$. The independent architecture improves only from $89.0\%$ to $89.8\%$, whereas ShareRes rises from $89.2\%$ to $93.6\%$, widening its advantage from $0.2$ to $3.8$ points. Because the amount of $F$ supervision is unchanged, the additional improvement cannot be explained by more target-configuration data. Instead, the widening gap indicates that ShareRes turns demonstrations collected under another sensor configuration into useful supervision for the target configuration, directly supporting its intended cross-configuration sharing mechanism.

**Sensor Utility.** Fig. 2 shows that the selected subset changes within a single successful rollout: UtilSelect uses no auxiliary sensor while observing, selects Depth during approach, combines Depth and Force during placement and release, and retains Force while closing the door. Rather than favoring one fixed configuration, the selected subset follows the changing interaction state. This behavior supports the distinction between sensors that are physically available and sensors that are useful for the current action chunk, which motivates observation-dependent selection in UtilSelect.

### 4.4 Ablation Studies

**Component Ablation.** Table 6 ablates modality dropout, ShareRes, and SetFiLM while disabling UtilSelect to isolate the configuration-compatible policy architecture. Modality dropout provides the basic ability to operate across varying sensor sets, improving the average from $94.75\%$ to $95.60\%$, but exposure to different configurations alone is not sufficient. Adding ShareRes or SetFiLM further raises the average to $96.05\%$ and $96.10\%$, respectively, while combining both reaches $96.60\%$. This pattern is consistent with the two modules addressing complementary requirements: ShareRes enables transferable sensory representations, whereas SetFiLM explicitly conditions action computation on the active sensor set.

**Adaptive Sensor Selection.** Table 7 isolates UtilSelect while keeping the trained policy fixed. All selection strategies in Table 7 use the same trained FlexSense checkpoint, whereas Table 6 reports separately trained component-ablation variants. UtilSelect raises the average success rate from $96.40\%$ when always using all available sensors and $96.67\%$ with the best fixed subset to $97.12\%$. Thus, additional sensors are not uniformly beneficial, nor is one subset optimal throughout a trajec-

Table 7: **Effect of adaptive sensor selection** on LIBERO (Liu et al., 2023) (§4.4). Each entry is the average success rate (%) over the four LIBERO suites.

| Method | $D$ | $F$ | $D + F$ | Avg. |
|---|---|---|---|---|
| All available sensors | 96.20 | 96.40 | 96.60 | 96.40 |
| Best fixed subset | 96.50 | 96.60 | 96.90 | 96.67 |
| UtilSelect | **96.95** | **97.00** | **97.40** | **97.12** |

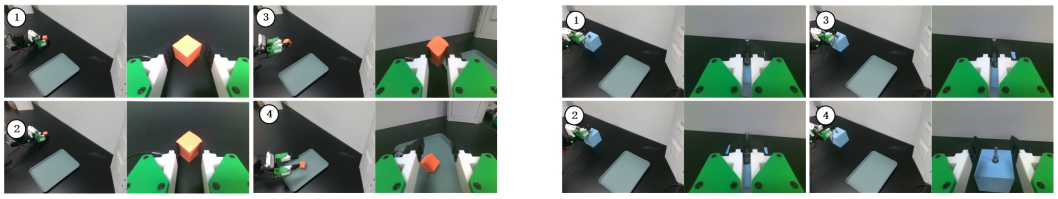

Figure 3: **Real-robot tasks** (§4.5). Left: **Pick-and-Place**. Right: **Precision Plug Insertion**. Each numbered panel shows one execution stage with third-person and wrist views.

Table 8: **Real-robot evaluation** under different sensor configurations. Each setting is evaluated over 20 trials per task (§4.5). Score denotes task progress for Pick-and-Place and time-normalized completion efficiency for Precision Plug Insertion, both on a 0–100 scale; SR denotes success rate (%). $D$ denotes Depth, $T$ denotes Tactile, and $\varnothing$ denotes no auxiliary sensor.

| Policy | Sensors | Pick-and-Place | | Precision Plug Insertion | |
|---|---|---|---|---|---|
| | | Score | SR | Score | SR |
| $\pi_{0.5}$ | $\varnothing$ | 93.0 | 80 | 7.5 | 20 |
| FlexSense | $\varnothing$ | 96.0 | 85 | 8.7 | 20 |
| FlexSense | $D$ | 99.0 | 95 | 13.8 | 30 |
| FlexSense | $T$ | 97.0 | 90 | 44.4 | 65 |
| FlexSense | $D + T$ | **100.0** | **100** | **65.3** | **80** |

tory. Together with the stage-wise behavior in Fig. 2, these results support the central assumption behind UtilSelect that sensor utility is both subset-dependent and observation-dependent.

### 4.5 Real-Robot Experiments

We evaluate the FlexSense policy under multiple sensing configurations on hardware. We consider Pick-and-Place, which primarily requires geometric localization and transport, and Precision Plug Insertion, which additionally requires fine alignment and contact-sensitive control. Representative executions are shown in Fig. 3, and quantitative results are reported in Table 8. Each policy–sensor configuration is evaluated over 20 trials per task. Besides binary success rate (SR), we report a normalized task score from 0 to 100; detailed scoring criteria are provided in Appendix H.1. The real-robot results reproduce the configuration-dependent sensing behavior observed in simulation with physical sensors. Depth primarily benefits Pick-and-Place, increasing SR from $85\%$ to $95\%$, whereas tactile sensing has a larger effect on Precision Plug Insertion, increasing SR from $20\%$ to $65\%$. Combining both reaches $100\%$ and $80\%$ SR, respectively. These results show that FlexSense does not merely tolerate changing sensor sets: the same policy can exploit complementary physical information from different sensors without configuration-specific retraining.

## 5 Conclusions

We introduced *FlexSense*, a unified Vision-Language-Action framework for robotic manipulation under heterogeneous sensor configurations. FlexSense combines ShareRes, SetFiLM, and UtilSelect, enabling a single policy to share knowledge across sensing setups, adapt its action generation to the available sensor set, and perform utility-guided sensor selection. Experiments on LIBERO, ManiSkill, and real-world manipulation demonstrate the effectiveness of this unified formulation across multiple sensor configurations. By treating sensor configuration as part of the policy context rather than a fixed property of the robot, FlexSense provides a flexible approach toward generalist robot policies with heterogeneous and evolving sensory interfaces.

### AI USE STATEMENT

Generative AI tools were used to assist with literature discovery, summarization of existing research, and limited language polishing during manuscript preparation. The authors independently examined the original sources and verified all cited literature, factual statements, and technical claims. All methodological choices, experimental settings, analyses, and conclusions were determined by the authors. The authors reviewed all AI-assisted content and take full responsibility for the final manuscript.

### REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the FlexSense architecture, training objectives, and sensor-selection procedure in Sec. 3. Experimental setups and primary evaluation protocols are described in Sec. 4.1. Additional implementation details and training hyperparameters are provided in Appendix A.1 and Appendix C. Additional configuration-level results and ablations are reported in Appendix E and Appendix G, while the complete real-robot protocol and scoring criteria are provided in Appendix H and Appendix H.1. A reproducibility checklist is provided in Appendix I. We will also release the implementation and configuration files necessary to reproduce the reported experiments.

## REFERENCES

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *CoRL*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. $\pi_0$: A Vision-Language-Action Flow Model for General Robot Control. In *RSS*, Los Angeles, CA, USA, June 2025b.

Yizhou Chen, Mark Van der Merwe, Andrea Sipos, and Nima Fazeli. Visuo-tactile transformers for manipulation. In *CoRL*, volume 205 of *Proceedings of Machine Learning Research*, pp. 2026–2040. PMLR, 14–18 Dec 2023.

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

Ria Doshi, Homer Rich Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. In *CoRL*, volume 270 of *Proceedings of Machine Learning Research*, pp. 496–512. PMLR, 06–09 Nov 2025.

Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. RH20T: A comprehensive robotic dataset for learning diverse skills in one-shot. In *ICRA*, pp. 653–660, 2024.

Ruoxuan Feng, Di Hu, Wenke Ma, and Xuelong Li. Play to the score: Stage-guided dynamic multi-sensory fusion for robotic manipulation. In *CoRL*, volume 270 of *Proceedings of Machine Learning Research*, pp. 340–363. PMLR, 06–09 Nov 2025.

Rohit Girdhar, Mannat Singh, Nikhila Ravi, Laurens van der Maaten, Armand Joulin, and Ishan Misra. Omnivore: A single model for many visual modalities. In *CVPR*, pp. 16102–16112, 2022.

Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. ImageBind: One embedding space to bind them all. In *CVPR*, pp. 15180–15190, June 2023.

Zihao He, Hongjie Fang, Jingjing Chen, Hao-Shu Fang, and Cewu Lu. Foar: Force-aware reactive policy for contact-rich robotic manipulation. *IEEE Robotics and Automation Letters*, 10(6):5625–5632, 2025.

Noriaki Hirose, Catherine Glossop, Dhruv Shah, and Sergey Levine. OmniVLA: An omni-modal vision-language-action model for robot navigation. *arXiv preprint arXiv:2509.19480*, 2025.

Pengbo Hu, Xingyu Li, and Yi Zhou. SHAPE: An Unified Approach to Evaluate the Contribution and Cooperation of Individual Modalities. In *IJCAI*, pp. 3064–3070, 2022.

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.

Physical Intelligence, Bo Ai, Ali Amin, R Aniceto, A Balakrishna, G Balke, K Black, G Bokinsky, S Cao, T Charbonnier, et al. $\pi_{0.7}$: A steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

Joshua Jones, Oier Mees, Carmelo Sferrazza, Kyle Stachowicz, Pieter Abbeel, and Sergey Levine. Beyond sight: Finetuning generalist robot policies with heterogeneous sensors via language grounding. In *ICRA*, pp. 5961–5968, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *CoRL*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025.

Zihan Lan, Weixin Mao, Haosheng Li, Le Wang, Tiancai Wang, Haoqiang Fan, and Osamu Yoshie. BFA: Best-feature-aware fusion for multi-view fine-grained manipulation. *IEEE Robotics and Automation Letters*, 10(9):8930–8937, 2025.

Yi-Lun Lee, Yi-Hsuan Tsai, Wei-Chen Chiu, and Chen-Yu Lee. Multimodal prompting with missing modalities for visual recognition. In *CVPR*, pp. 14943–14952, June 2023.

Chengmeng Li, Junjie Wen, Yaxin Peng, Yan Peng, and Yichen Zhu. PointVLA: Injecting the 3D world into vision-language-action models. *IEEE Robotics and Automation Letters*, 11(3):2506–2513, 2026.

Hao Li, Yizhi Zhang, Junzhe Zhu, Shaoxiong Wang, Michelle A Lee, Huazhe Xu, Edward Adelson, Li Fei-Fei, Ruohan Gao, and Jiajun Wu. See, hear, and feel: Smart sensory fusion for robotic manipulation. In *CoRL*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1368–1378. PMLR, 14–18 Dec 2023.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. CogACT: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

Jason Jingzhou Liu, Yulong Li, Kenneth Shaw, Tony Tao, Russ Salakhutdinov, and Deepak Pathak. FACTR: Force-Attending Curriculum Training for Contact-Rich Policy Learning. In *RSS*, Los Angeles, CA, USA, June 2025.

Xingyuming Liu, Ruichun Ma, Heyu Guo, Qixiu Li, Qingwen Yang, Lin Luo, Shiqi Jiang, Chenren Xu, Jiaolong Yang, and Baining Guo. MuseVLA: An adaptive multimodal sensing vision-language-action model for robotic manipulation. *arXiv preprint arXiv:2606.17598*, 2026.

Mengmeng Ma, Jian Ren, Long Zhao, Davide Testuggine, and Xi Peng. Are multimodal transformers robust to missing modality? In *CVPR*, pp. 18177–18186, June 2022.

Natalia Neverova, Christian Wolf, Graham Taylor, and Florian Nebout. Moddrop: Adaptive multi-modal gesture recognition. *IEEE TPAMI*, 38(8):1692–1706, 2016.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Jim Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *RSS*, Delft, Netherlands, 2024.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, 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. *arXiv preprint arXiv:2310.08864*, 2025.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. FiLM: Visual Reasoning with a General Conditioning Layer. In *AAAI*, pp. 3942–3951, 2018.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient Action Tokenization for Vision-Language-Action Models. In *RSS*, Los Angeles, CA, USA, June 2025.

Jens Piekenbrinck, Alexander Hermans, Narunas Vaskevicius, Timm Linder, and Bastian Leibe. RGB-D cube R-CNN: 3D object detection with selective modality dropout. In *CVPR Workshops*, pp. 1997–2006, June 2024.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Jiayuan Gu, Zhigang Wang, Yan Ding, Bin Zhao, Dong Wang, and Xuelong Li. SpatialVLA: Exploring Spatial Representations for Visual-Language-Action Models. In *RSS*, Los Angeles, CA, USA, June 2025.

Skand Skand, Bikram Pandit, Chanho Kim, Li Fuxin, and Stefan Lee. Simple masked training strategies yield control policies that are robust to sensor failure. In *CoRL*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4463–4482. PMLR, 06–09 Nov 2025.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-Kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Demonstrating GPU Parallelized Robot Simulation and Rendering for Generalizable Embodied AI with ManiSkill3. In *RSS*, Los Angeles, CA, USA, June 2025.

Hu Wang, Yuanhong Chen, Congbo Ma, Jodie Avery, Louise Hull, and Gustavo Carneiro. Multi-modal learning with missing modality via shared-specific feature modelling. In *CVPR*, pp. 15878–15887, June 2023.

Qi Wei, Pengwen Xiong, Aiguo Song, and Qiang Li. Task-specific embodied tactile sensing for dexterous hand. In *ICRA*, pp. 11825–11830, 2025a.

Xiangyi Wei, Haotian Zhang, Xinyi Cao, Siyu Xie, Weifeng Ge, Yang Li, and Changbo Wang. Audio-VLA: Adding contact audio perception to vision-language-action model for robotic manipulation. *arXiv preprint arXiv:2511.09958*, 2025b.

Junjie Wen, Yichen Zhu, Minjie Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Xiaoyu Liu, Chaomin Shen, Yaxin Peng, and Feifei Feng. DiffusionVLA: Scaling robot foundation models via unified diffusion and autoregression. In *ICML*, volume 267 of *Proceedings of Machine Learning Research*, pp. 66558–66574. PMLR, 13–19 Jul 2025.

Zequn Yang, Yake Wei, Ce Liang, and Di Hu. Quantifying and enhancing multi-modal robustness with modality preference. In *ICLR*, volume 2024, pp. 15860–15872, 2024.

Jiawen Yu, Hairuo Liu, Qiaojun Yu, Jieji Ren, Ce Hao, Haitong Ding, Guangyu Huang, Guofan Huang, Yan Song, Panpan Cai, Wenqiang Zhang, and Cewu Lu. ForceVLA: Enhancing VLA models with a force-aware MoE for contact-rich manipulation. In *NeurIPS*, volume 38, pp. 93409–93439. Curran Associates, Inc., 2025.

Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Zhuoguang Chen, Tao Jiang, and Hang Zhao. Depthvla: Enhancing vision-language-action models with depth-aware spatial reasoning. *arXiv preprint arXiv:2510.13375*, 2025.

Sukwon Yun, Inyoung Choi, Jie Peng, Yangfan Wu, Jingxuan Bao, Qiyiwen Zhang, Jiayi Xin, Qi Long, and Tianlong Chen. Flex-MoE: Modeling arbitrary modality combination via the flexible mixture-of-experts. In *NeurIPS*, volume 37, pp. 98782–98805. Curran Associates, Inc., 2024.

Jiajun Zhai, Hao Shi, Shangwei Guo, Kailun Yang, and Kaiwei Wang. E-VLA: Event-augmented vision-language-action model for dark and blurred scenes. *arXiv preprint arXiv:2604.04834*, 2026.

Haozhe Zhang, Sixian Li, Yifei Zhang, Zezheng Huai, Hao Chen, Chunhua Shen, Jingjing Gong, and Xipeng Qiu. CoRE-VLA: Towards scalable and robust vision-language-action modeling via conditional routing of experts. *arXiv preprint arXiv:2607.03693*, 2026.

Zongzheng Zhang, Haobo Xu, Zhuo Yang, Chenghao Yue, Zehao Lin, Huan-ang Gao, Ziwei Wang, and Hao Zhao. Elucidating the design space of torque-aware vision-language-action models. In *CoRL*, volume 305 of *Proceedings of Machine Learning Research*, pp. 4019–4037. PMLR, 27–30 Sep 2025.

Yichen Zhu, Zhicai Ou, Feifei Feng, and Jian Tang. Any2Policy: Learning visuomotor policy with any-modality. In *NeurIPS*, volume 37, pp. 133518–133540. Curran Associates, Inc., 2024.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *CoRL*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023.

# Supplementary Material for FlexSense

This supplementary material provides additional implementation details, complete experimental protocols, cross-configuration analyses, reproducibility information, and a scoped discussion of limitations and future work for the main paper.

## A Experiment Setting

### A.1 Software and Hardware Environment

**Implementation Details.** FlexSense extends existing VLA training settings rather than introducing a separate action-model implementation. For LIBERO, we initialize from the StarVLA-$\pi$ LIBERO-4in1 checkpoint. For ManiSkill and the real-robot experiments, we initialize from $\pi_{0.5}$. Table A1 summarizes the software and hardware environment used in our experiments.

Table A1: **Computing environment** used by the training and evaluation protocols (§A.1).

| Component | Version or setting | Role |
|---|---|---|
| GPU | 8× NVIDIA RTX 4090 (24 GB each) | Distributed training and parallel evaluation |
| Operating system | Ubuntu 22.04.4 LTS | Training, inference, and simulator host |
| Python | 3.10.20 | Runtime environment |
| PyTorch | 2.6.0 + CUDA 12.4 | FlexSense/StarVLA training and inference |
| Torchvision | 0.21.0 + CUDA 12.4 | Image preprocessing and visual input support |
| Transformers | 4.57.0 | Qwen3-VL backbone and multimodal processing |
| Accelerate | 1.5.2 | Distributed training orchestration |
| DeepSpeed | 0.16.9 | ZeRO-3 distributed optimization |
| LIBERO | LIBERO 0.1.0 | Task definitions and benchmark interface |
| LIBERO simulator | robosuite 1.4.0 / MuJoCo 2.3.7 | LIBERO simulation and rollout evaluation |
| ManiSkill simulator | ManiSkill 3.0.1 / SAPIEN | ManiSkill demonstration replay and evaluation |
| $\pi_{0.5}$ backend | PyTorch implementation | ManiSkill and real-robot policy initialization |
| Optimizer | AdamW | Parameter optimization |
| Numerical precision | BF16 training and inference | Memory-efficient model execution |
| Distributed strategy | DeepSpeed ZeRO-3 | Multi-GPU parameter and optimizer-state sharding |
| LIBERO evaluation seed | 7 | Simulator initialization and rollout replication |

**Optimization.** We optimize all trainable components with AdamW. The base VLA and newly introduced FlexSense modules use separate parameter groups where applicable. Table A2 summarizes the optimization settings, training schedules, action horizons, configuration sampling, and utility-target parameters used for each experimental setting.

**Datasets and Model Resources.** We use LIBERO and ManiSkill for the simulation experiments, and we collect the real-robot data in-house through VR-controller teleoperation. We use the upstream datasets, simulators, codebases, and model checkpoints under their original terms. Because the repository license does not automatically determine the license of every downloaded asset or model weight, we separate code, assets, checkpoints, and in-house data in Table A3. In the code release, we will retain all required notices and attribution.

## B Benchmarks and Sensor Configurations

### B.1 Common Observation and Action Interface

RGB observations, language instructions, and robot proprioception are always available. Let $\mathcal{M}$ denote the set of optional physical modalities and $C \subseteq \mathcal{M}$ the sensor configuration available for a demonstration or deployment episode. At time $t$, the unified policy predicts an $H$-step action chunk,

$$A_t = [a_t, \ldots, a_{t+H-1}] \sim \pi_\theta(\cdot \mid o_t^C, C), \tag{6}$$

using one shared parameter set for every supported configuration. During online selection, UtilSelect may activate a subset $\widehat{S}_t \subseteq C$, but it never introduces a modality that is absent from $C$.

Table A2: **Training and selection hyperparameters** (§A.1).

| Hyperparameter | LIBERO | ManiSkill | Real robot |
|---|---|---|---|
| Base checkpoint | StarVLA-$\pi$ 4in1 | $\pi_{0.5}$ | $\pi_{0.5}$ |
| Optimizer | AdamW | AdamW | AdamW |
| Base-VLA learning rate | VLM frozen; Action DiT: $1 \times 10^{-5}$ | VLM frozen; Action Expert: $2.5 \times 10^{-5}$ | VLM frozen; Action Expert: $1 \times 10^{-5}$ |
| New-module learning rate | $5 \times 10^{-5}$ | $5 \times 10^{-5}$ | $5 \times 10^{-5}$ |
| Weight decay | $1 \times 10^{-8}$ | $1 \times 10^{-2}$ | $1 \times 10^{-8}$ |
| Global batch size | 480 | 128 | 256 |
| Training updates | 1,600 updates | 2,000 updates | 1,350 updates |
| Warm-up / scheduler | 100 updates / cosine decay | 100 updates / cosine decay | 50 updates / cosine decay |
| Action-chunk horizon $H$ | 8 | 50 | 50 |
| Configuration sampling | Uniform over subsets of $\{\varnothing, D, F, D{+}F\}$ | Uniform over subsets of $\{\varnothing, D, F, D{+}F\}$ | Uniform over subsets of $\{\varnothing, D, T, D{+}T\}$ |
| Utility repeats $K$ | 4 | 4 | 4 |
| Utility constant $\epsilon$ | $1 \times 10^{-6}$ | $1 \times 10^{-6}$ | $1 \times 10^{-6}$ |
| Training seed | 42 | 1000 | 1000 |

Table A3: **Dataset assets and model resources** used in the experiments (§A.1).

| Asset | Source / license |
|---|---|
| LIBERO | Public benchmark; code: MIT |
| ManiSkill 3 | Code: Apache-2.0; assets: CC BY-NC 4.0 |
| robosuite / MuJoCo | Public simulator releases |
| SAPIEN | Public simulator release |
| StarVLA | Public release; code: MIT |
| $\pi_{0.5}$ / openpi | Code: Apache-2.0; additional component terms |
| Real-robot demonstrations | In-house collection |
| RealSense D436 streams | In-house collection |
| Xense Photon streams | In-house collection |

### B.2 LIBERO Protocol

We evaluate on the Spatial, Object, Goal, and Long suites of LIBERO, each containing 10 tasks and 50 demonstrations per task. The resulting training corpus contains 2,000 demonstrations before modality masking. We initialize from the StarVLA-$\pi$ LIBERO-4in1 checkpoint and jointly fine-tune one policy across all four suites. During replay in robosuite/MuJoCo, metric depth is reconstructed from the agent-view and wrist-view depth buffers. The optional force observation is an uncompensated six-dimensional wrist wrench in the local sensor frame, $w_t = [F_x, F_y, F_z, \tau_x, \tau_y, \tau_z]$. Each task is evaluated over 50 rollouts, which gives 500 rollouts per suite and 2,000 rollouts per sensor configuration.

### B.3 ManiSkill Protocol

We evaluate on PegInsertionSide, PickCube, StackCube, PlugCharger, and PushCube. One $\pi_{0.5}$-initialized policy is jointly trained on 1,000 successful official motion-planning demonstrations per task, for 5,000 demonstrations in total. Metric depth is rendered by ManiSkill 3.0.1/SAPIEN, and the six-dimensional simulated wrist wrench is expressed in the tool-center-point (TCP) frame. Expert actions are converted from `pd_joint_pos` control to seven-dimensional `pd_ee_delta_pose` actions before training. Each task is evaluated over 200 episodes.

### B.4 Real-Robot Protocol

The real-world platform is an ARX-X5 (2023) single-arm robot with a custom gripper. Two RealSense D436 RGB-D cameras provide a fixed third-person view and a wrist-mounted view. Two Xense Photon vision-based tactile sensors are mounted on the gripper fingertips. FlexSense is ini-

tialized from $\pi_{0.5}$ and evaluated on Pick-and-Place and Precision Plug Insertion. Demonstrations are collected through VR-controller teleoperation. In contrast to the simulation setup, the optional contact modality is Tactile (T), not Force (F).

# C Additional Implementation Details

## C.1 Shared-Residual Sensory Encoding

For each optional modality $m \in \mathcal{M}$, a modality-specific stem $E_m$ maps the raw observation $x_t^m$ to $N_m$ tokens of width $d_s$. Image-like modalities, such as depth and vision-based tactile streams, use convolutional stems, whereas low-dimensional wrench signals use an MLP stem. The token count may differ across modalities. After modality-specific layer normalization, the same two-layer Transformer $B_{\mathrm{sh}}$ processes every optional modality independently:

$$S_t^m = B_{\mathrm{sh}}(\mathrm{LN}_m(T_t^m)), \qquad R_t^m = A_m(\mathrm{LN}_m(T_t^m)), \tag{7}$$

$$\Delta_t^m = \gamma_m R_t^m, \qquad Z_t^m = P_m(S_t^m + \Delta_t^m) + e_m. \tag{8}$$

Here, $A_m$ is a modality-specific bottleneck adapter, $P_m$ projects to the action-policy width, and $e_m$ is a sensor-type embedding. The scalar $\gamma_m$ is initialized to zero. Thus, each new branch begins as a shared representation with a zeroed modality-specific correction, which reduces disruption to the pretrained VLA at initialization.

The active residuals are regularized by

$$\mathcal{L}_{\mathrm{res}} = \sum_{m \in \mathcal{M}} \frac{\sum_b a_{b,m} \,\mathrm{mean}[(\Delta_b^m)^2]}{\sum_b a_{b,m} + \epsilon}, \qquad \mathcal{L} = \mathcal{L}_{\mathrm{act}} + 10^{-4}\mathcal{L}_{\mathrm{res}}, \tag{9}$$

where $a_{b,m} = \mathbf{1}[m \in \widetilde{C}_b]$ indicates whether modality $m$ remains active for sample $b$ after configuration-constrained modality dropout. For the flow-matching instantiations in this paper, $\mathcal{L}_{\mathrm{act}} = \mathcal{L}_{\mathrm{flow}}$.

## C.2 Sensor-Set-Conditioned FiLM

Let $c_t \in \{0,1\}^{|\mathcal{M}|}$ encode the active optional-sensor subset, with $c_t[m] = \mathbf{1}[m \in S_t]$, while $C$ denotes the physically available optional-sensor set. A configuration-dependent attention mask assigns $-\infty$ logits to tokens outside the active subset. Masking determines what information is visible; SetFiLM additionally changes how the shared action block processes the visible set. For action block $l$,

$$\widetilde{H}_{t,l} = \mathrm{LN}(H_{t,l}) \odot [1 + s_l(\tau) + \gamma_l(c_t)] + b_l(\tau) + \beta_l(c_t), \tag{10}$$

where $s_l(\tau)$ and $b_l(\tau)$ are inherited from the base action expert. The configuration-conditioned scale $\gamma_l(c_t)$ and shift $\beta_l(c_t)$ are produced by a lightweight MLP and are zero initialized, making SetFiLM a no-op at initialization. The binary configuration passed to SetFiLM must match the subset actually exposed by the attention mask; at online inference, this is the selected subset $\widehat{S}_t$, not merely the physically installed set $C$.

## C.3 Configuration-Constrained Modality Dropout

For a training demonstration with native configuration $C_i$, we sample a feasible training subset $\widetilde{C}_i \subseteq C_i$. Modalities outside $C_i$ are never synthesized or imputed. The shared policy is optimized under $\widetilde{C}_i$ using the native action objective and the residual penalty. This distinction is important for heterogeneous datasets: modality dropout can remove an observed sensor, but it cannot convert a depth-only demonstration into a force-supervised demonstration. For each training sample, we draw $\widetilde{C}_i$ uniformly from the feasible subsets of its native configuration $C_i$; thus the empty optional set remains in the sampling support whenever it is feasible. The sampled subset is applied independently per sample and is used consistently by both the attention mask and SetFiLM.

### C.4 Utility-Guided Sensor Selection

**Offline Targets.** After unified-policy training, we freeze $\pi_\theta$. For a demonstration with available configuration $C$, each candidate subset $S \subseteq C$ is evaluated against the expert action chunk. Within repeat $k$, all subsets use the same initial flow noise, reducing variance in pairwise comparisons. The action error and set-conditional marginal utility are

$$E_t(S) = \frac{1}{K} \sum_{k=1}^{K} \mathrm{SmoothL1}(\widehat{A}^S_{t,k}, A^\star_t), \tag{11}$$

$$u^\star_{m,t}(S) = \frac{E_t(S) - E_t(S \cup \{m\})}{E_t(S) + E_t(S \cup \{m\}) + \epsilon}, \qquad m \in C \setminus S. \tag{12}$$

SmoothL1 is averaged over all action dimensions and time steps in the predicted action chunk. The target measures teacher-defined improvement in imitation error and must not be interpreted as causal sensor importance.

**Scorer Distillation.** Frozen RGB/vision-language, proprioceptive, and optional-sensor representations are pooled into source-level summaries and augmented with source-type embeddings. A candidate-subset embedding queries the summaries through a small cross-attention module. An MLP with Softplus output predicts the non-negative error $\widehat{E}_t(S) = g_\phi(o^C_t, C, S)$ and is trained with SmoothL1 regression while the VLA remains frozen.

**Online Selection.** At every action-policy query,

$$\widehat{S}_t = \arg\min_{S \subseteq C} \widehat{E}_t(S), \qquad \widehat{A}_t = \pi_\theta(\cdot \mid o_t^{\widehat{S}_t}, \widehat{S}_t). \tag{13}$$

For the two optional sensors used in each experiment, exhaustive scoring requires at most four candidate subsets. The selection is recomputed once per action chunk, so the active sensor subset may change across manipulation stages.

## D Evaluation Protocols and Metrics

### D.1 Success Aggregation

Task success is the primary metric. Let $\mathrm{SR}_{q,C} = 100 N^{\mathrm{succ}}_{q,C} / N^{\mathrm{roll}}_{q,C}$ denote the success rate for task $q$ under configuration $C$. We report the task average, configuration average, and worst-configuration performance as

$$\overline{\mathrm{SR}}_C = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathrm{SR}_{q,C}, \qquad \overline{\mathrm{SR}} = \frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} \overline{\mathrm{SR}}_C, \tag{14}$$

$$\mathrm{WorstSR} = \min_{C \in \mathcal{C}} \overline{\mathrm{SR}}_C. \tag{15}$$

For a unified policy and its configuration-specific specialist, the signed gap is

$$\Delta_C = \overline{\mathrm{SR}}^{\mathrm{unified}}_C - \overline{\mathrm{SR}}^{\mathrm{specialist}}_C. \tag{16}$$

Positive values favor FlexSense. Each benchmark uses the fixed training seed listed in Table A2. Reported success rates are aggregated over the rollout counts specified in Appendix B; no cross-seed variability is reported.

## E Additional Configuration-Level Results

### E.1 LIBERO Specialist Gap

Table A4 reorganizes the reported LIBERO results to show the signed difference between the unified policy and the matching specialist. The largest configuration-average absolute gap is 1.45 percentage points. Averaged uniformly across the four configurations, the specialists obtain 96.80% and the single FlexSense policy obtains 97.14%, a signed difference of +0.34 points. These aggregates are derived from the values in the main paper and do not replace the full suite-level table.

Table A4: **Unified-minus-specialist success-rate gap** on LIBERO (§E). Values are percentage points; positive values favor the unified FlexSense policy.

| Sensors | Spatial | Object | Goal | Long | Average |
|---|---|---|---|---|---|
| ∅ | −0.40 | +1.40 | +0.20 | +4.60 | +1.45 |
| D | −0.40 | 0.00 | +0.20 | −0.40 | −0.15 |
| F | −0.20 | +0.20 | +0.40 | −0.60 | −0.05 |
| D+F | −1.40 | +0.60 | +0.40 | +0.80 | +0.10 |

### E.2 ManiSkill Configuration Breakdown

For completeness, Table A5 preserves the full task-by-configuration layout used to test whether configuration compatibility extends beyond LIBERO. The specialist and unified models use the same 5,000-demonstration corpus, action conversion, evaluation seeds, and number of rollouts.

Table A5: **Specialist-versus-unified comparison on ManiSkill** (§E). All entries are task success rates (%). D denotes Depth, F denotes Force, and ∅ denotes no auxiliary sensor.

| Policy | Sens. | Peg Insert. | Pick Cube | Stack Cube | Plug Charge. | Push Cube | Average |
|---|---|---|---|---|---|---|---|
| ∅ specialist | ∅ | 7.5 | 86.0 | 67.0 | 2.5 | 100 | 52.6 |
| D specialist | D | 11.0 | 95.0 | 75.0 | 6.0 | 100 | 57.4 |
| F specialist | F | 15.5 | 90.5 | 72.5 | 12.5 | 100 | 58.2 |
| D+F specialist | D+F | 16.0 | 96.0 | 76.5 | 13.5 | 100 | 60.4 |
| FlexSense | ∅ | 8.5 | 87.0 | 68.0 | 3.0 | 99.0 | 53.1 |
| FlexSense | D | 10.5 | 93.5 | 73.5 | 5.0 | 100 | 56.5 |
| FlexSense | F | 13.5 | 90.0 | 70.5 | 10.5 | 100 | 56.9 |
| FlexSense | D+F | 14.0 | 95.0 | 75.0 | 12.0 | 100 | 59.2 |

## F Cross-Configuration Data Utilization

We further detail the controlled LIBERO-10 protocol summarized in the main paper. The target configuration is Force (F): the amount of F data is fixed, and every model is evaluated under F. We then add separately collected Depth (D) demonstrations at 10%, 50%, and 100% coverage. This setting differs from ordinary multi-task training because the auxiliary demonstrations do not contain the target force observation. An improvement under F therefore measures whether the shared architecture transfers task and control structure from D demonstrations rather than simply receiving additional target-configuration examples.

To isolate cross-configuration transfer, the trajectory partition, F-data pool, evaluation episodes, initialization, checkpoint rule, and optimization budget are fixed across the Independent and ShareRes variants. D trajectories are nested across coverage levels: the 10% split is contained in the 50% split, which is contained in the 100% split.

Table A6: **Cross-configuration data utilization on LIBERO-10** (§F). The F pool is fixed while the coverage of additional D data varies. All methods are evaluated under F.

| Architecture | D coverage | Successes | Success rate (%) |
|---|---|---|---|
| Independent | 10% | 445 | 89.0 |
| Independent | 50% | 447 | 89.4 |
| Independent | 100% | 449 | 89.8 |
| ShareRes | 10% | 446 | 89.2 |
| ShareRes | 50% | 457 | 91.4 |
| ShareRes | 100% | 468 | 93.6 |

For each coverage $r$, we define the transfer gain as

$$G(r) = \mathrm{SR}_{\mathrm{ShareRes}}(r) - \mathrm{SR}_{\mathrm{Independent}}(r). \tag{17}$$

Reporting $G(r)$ together with absolute success rates prevents a large relative gain at a low baseline from being overinterpreted.

# G Additional Ablations and Diagnostics

## G.1 ShareRes and SetFiLM

The component ablation begins with a unified policy without modality dropout, adds configuration-constrained dropout, and then tests ShareRes and SetFiLM. The most informative comparison is not only the configuration average but also the worst configuration: a component may improve the mean while leaving one sensing setup brittle.

Table A7: **Component ablation on LIBERO** (§G). Each entry is the average success rate (%) over Spatial, Object, Goal, and Long.

| Variant | ∅ | D | F | D+F | Average | Worst |
|---|---|---|---|---|---|---|
| No modality dropout | 94.2 | 94.8 | 94.6 | 95.4 | 94.75 | 94.2 |
| Modality dropout | 96.1 | 95.3 | 95.2 | 95.8 | 95.60 | 95.2 |
| + ShareRes | 96.2 | 95.9 | 95.8 | 96.3 | 96.05 | 95.8 |
| + SetFiLM | 96.5 | 95.8 | 95.7 | 96.4 | 96.10 | 95.7 |
| + ShareRes + SetFiLM | 97.2 | 96.2 | 96.4 | 96.6 | 96.60 | 96.2 |

## G.2 Utility-Based Selection

We compare three sensor-selection strategies using the same fixed trained FlexSense checkpoint: all available sensors, a best fixed subset, and the learned UtilSelect scorer. The best-fixed-subset strategy uses one subset throughout a rollout, whereas UtilSelect may change the active subset at each action-policy query. To quantify efficiency, we additionally report the average number of optional sensors retained per query and the online selection overhead.

Table A8: **Adaptive sensor utilization on LIBERO** (§G). Success rates are averaged over the four suites. Sens./query denotes the number of optional sensors passed to the action policy at each query.

| Method | D | F | D+F | Average | Sens./query | Selection overhead (ms) |
|---|---|---|---|---|---|---|
| All available | 96.20 | 96.40 | 96.60 | 96.40 | 1.33 | 0.00 |
| Best fixed subset | 96.50 | 96.60 | 96.90 | 96.67 | 1.00 | 0.00 |
| UtilSelect | 96.95 | 97.00 | 97.40 | 97.12 | 1.03 | 2.70 |

# H Real-Robot Evaluation Details

The real-robot study evaluates one shared FlexSense policy under ∅, D, T, and D+T. Each task is trained from 70 VR-teleoperated demonstrations, and each policy–sensor configuration is evaluated over 20 trials. The $\pi_{0.5}$ baseline is fine-tuned on the same 70 demonstrations per task and evaluated on the same 20 initial-state trials and evaluation horizon as FlexSense, but receives no optional depth or tactile input. Actions are executed at 15 Hz, while RGB-D and tactile observations are acquired at 30 Hz and aligned to each policy query by nearest timestamp. Each trial starts from a randomized feasible object pose, and the policy predicts action chunks with horizon $H = 50$. A trial is counted as successful only when the task-specific terminal condition is reached within the evaluation horizon.

## H.1 Real-Robot Scoring Criteria

**Pick-and-Place score.** We use a stage-based task-progress score to capture partial completion beyond binary success. Each trial receives the score of the highest completed stage: 0 if the robot fails to establish meaningful interaction with the target object; 20 after reaching the object and establishing grasp contact; 40 after achieving a stable grasp; 60 after lifting the object clear of its

Table A9: **Real-robot evaluation under different sensor configurations** (§H). Each task reports a task-specific score and success rate (SR, %). D denotes Depth, T denotes Tactile, and ∅ denotes no auxiliary sensor.

| Policy | Sens. | Pick-and-Place | | Precision Plug Insertion | |
|---|---|---|---|---|---|
| | | Score | SR | Score | SR |
| $\pi_{0.5}$ | ∅ | 93.0 | 80 | 7.5 | 20 |
| FlexSense | ∅ | 96.0 | 85 | 8.7 | 20 |
| FlexSense | D | 99.0 | 95 | 13.8 | 30 |
| FlexSense | T | 97.0 | 90 | 44.4 | 65 |
| FlexSense | D+T | 100.0 | 100 | 65.3 | 80 |

Table A10: **Real-robot protocol** for the physical experiments (§H).

| Field | Setting | Purpose |
|---|---|---|
| Demonstrations per task | 70 | Training-data scale |
| Trials per task/configuration | 20 | Statistical denominator |
| Control / observation rate | 15 Hz action execution; 30 Hz RGB-D / tactile acquisition | Temporal reproducibility |
| Action horizon $H$ | 50 | Predicted action-chunk length |
| Tactile synchronization | Nearest-timestamp alignment at each policy query | Multimodal temporal alignment |
| Initial-state randomization | Randomized object pose within a predefined feasible workspace | Trial diversity |
| Success and score rules | Task-specific progress score for Pick-and-Place; completion-time score for Plug Insertion | Metric reproducibility |

initial support; 80 after transporting the object to the target region without completing a stable final placement; and 100 after the object is successfully placed and released within the target region. The reported score is averaged over all 20 trials.

**Precision Plug Insertion score.** Binary success requires the plug to reach the predefined fully inserted state. To additionally measure insertion efficiency, failed trials are assigned zero score before applying the time-based rule. For a successful trial, let $t$ denote the elapsed time from the beginning of the insertion attempt, $T_{\text{fast}} = 5\,\text{s}$ denote the threshold for full efficiency score, and $T_{\max} = 30\,\text{s}$ denote the maximum allowed completion time. We compute

$$s_{\text{plug}} = \begin{cases} 100, & t \le T_{\text{fast}}, \\ 100\dfrac{T_{\max} - t}{T_{\max} - T_{\text{fast}}}, & T_{\text{fast}} < t < T_{\max}, \\ 0, & t \ge T_{\max}. \end{cases} \tag{18}$$

Thus, failed trials receive zero score, while faster successful insertions receive higher scores. The reported value is averaged over all 20 trials.

## I Reproducibility Checklist

Our code will be publicly released to ensure reproducibility. Table A11 summarizes the minimum release items.

## J Limitations and Future Work

**Limitations.** FlexSense is currently evaluated with at most two optional sensors in each setting. Consequently, exhaustive UtilSelect scoring remains inexpensive in our experiments, but its efficiency for substantially larger sensor arrays has not yet been verified; this limitation can be addressed with greedy or autoregressive subset selection. The experiments also cover multiple fixed sensor configurations but do not explicitly test abrupt sensor loss, delay, or corruption during an ongoing trajectory. Controlled mid-episode masking and corruption tests, together with dynamic modality-dropout training, provide a direct way to evaluate and improve this form of robustness.

Table A11: **Reproducibility artifacts and documentation.**

| Artifact | Contents |
|---|---|
| Environment specification | Python, PyTorch, CUDA, GPU, simulator, and package versions |
| Code provenance | Anonymous repository or supplementary code, experiment configurations, and implementation patches |
| Dataset manifest | Benchmark suite, task, trajectory identifier, and native sensor configuration |
| Data-split manifest | Policy-training trajectories and episode-level UtilSelect training, validation, and test splits |
| Resolved training configurations | Optimization hyperparameters, module switches, modality-dropout distributions, and trainable parameter groups |
| Checkpoint manifest | Base checkpoint, trained checkpoint, training seed, update count, and checkpoint hash |
| Evaluation manifest | Benchmark task, sensor configuration, rollout seed, initial-state identifier, and number of trials |
| Raw evaluation logs | Per-episode success, task score, selected sensor subset, and termination reason |
| Utility-target cache | Candidate sensor subsets, shared flow-noise seeds, expert actions, and counterfactual action errors |
| License and attribution | Licenses and attribution notices for code, datasets, simulator assets, and pretrained checkpoints |

**Social Impact.** By allowing one VLA checkpoint to operate across heterogeneous sensor configurations, FlexSense may reduce repeated model training and simplify the deployment of manipulation policies on robots with different sensing hardware. At the same time, RGB-D, force, and tactile streams may reveal private environments, human activity, or sensitive contact interactions, and errors in sensor selection or action prediction may propagate to unsafe robot motion, object damage, or failed task execution. Flexible sensor use should therefore not be interpreted as guaranteed reliability under sensor failure. Deployment in safety- or privacy-sensitive settings should use consent-aware data collection, appropriate access controls, sensor-health monitoring, conservative fallback behaviors, task-specific validation, and human oversight where consequences are substantial.