SUREFlow: State-space Uncertainty-aware REsidual Flow Matching for Robust Robot Manipulation

Anonymous Authors

***Abstract*— Generative vision-language-action policies have advanced robot manipulation, but they often exhibit instability under noise, partial observability, and stochastic initial conditions. During extended rollouts, small velocity errors accumulate, degrading execution reliability. Existing diffusion and flow-based policies typically assume homoscedastic residuals and lack explicit uncertainty modeling within action generation, limiting robustness during iterative rollout. We propose SUREFlow, a state-space uncertainty-aware residual flow matching framework built on a Mamba backbone. The method jointly predicts action velocities and input-dependent residual uncertainty, enabling selective refinement of unreliable action dimensions without environment feedback while preserving computational efficiency. On LIBERO, SUREFlow achieves 92.6% average success rate (SR), outperforming the Mamba-based MaIL by 34.3%. On LIBERO-PRO, it attains around 50% SR using only 179M parameters, achieving performance comparable to large VLAs with 3-7B parameters. Calibrated uncertainty with residual refinement enables scalable and stable extended-horizon rollout execution in generative robot control.**

I. Introduction

Learning robot manipulation (RM) policies from demonstrations has become a dominant paradigm, where recent vision-language-action (VLA) models leverage multi-view visual observations and natural language conditioning to enable generalizable robotic skills across diverse tasks [1], [2], [3]. Despite this progress, many RM approaches still rely on direct action regression formulations [4], which yield deterministic predictions and are known to struggle when demonstrations exhibit multimodal behaviors, as commonly observed in real-world tasks [5], [6].

To address these limitations, generative policy learning methods have recently gained attention. Diffusion-based and flow-based policies model the conditional distribution over actions rather than predicting a single output, enabling stochastic action generation and improved expressiveness, particularly in robotic manipulation settings [1], [7], [8]. Such methods have demonstrated improved robustness and flexibility compared to deterministic baselines. However, purely continuous generative policies typically operate at a single temporal scale and lack explicit temporal abstraction. As a result, stochasticity introduced during generation may accumulate over long action sequences, leading to unstable execution and inconsistent behaviors [6], [7]. Moreover, these approaches typically incur high computational costs due to iterative sampling and often rely on transformer-based architectures that increase computational cost largely.

In parallel, prior work has explored hierarchical and skill-based representations for RM, aiming to learn reusable skills

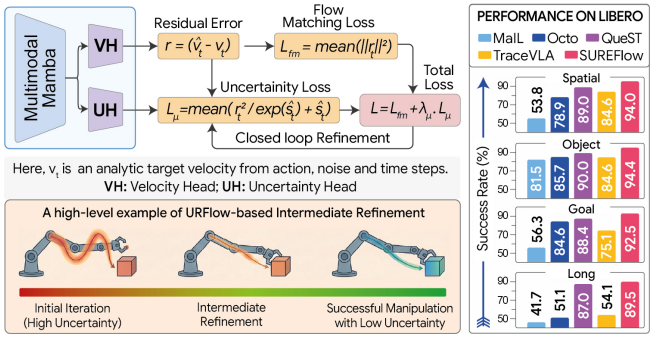

Fig. 1. Overview of URFlow, illustrating closed-loop refinement of uncertain action dimensions via internal residual updates during inference, without external feedback. The right panel shows representative LIBERO results, demonstrating consistent improvements over recent SOTA baselines.

directly from data rather than relying on hand-crafted primitives [9], [10], [11]. However, most such approaches either separate skill learning from low-level control or depend on explicit planning modules, limiting seamless integration with end-to-end generative policies. Meanwhile, alternative sequence modeling architectures have recently gained attention for long-horizon reasoning. State-space models such as Mamba [12] offer linear time complexity and strong temporal modeling capabilities, providing an efficient alternative to large backbones like Transformers. Despite their success in language and vision domains, their application to generative robot control remains limited. Moreover, generative manipulation policies introduce stochasticity during action generation, which can accumulate over long-horizon rollouts and lead to unstable execution when actions are applied open-loop. Existing approaches typically lack explicit mechanisms to quantify or correct uncertain action predictions during inference, limiting their robustness in complex manipulation tasks utilizing a comparatively lightweight architecture.

In this work, we address this gap by proposing *"State-space Uncertainty-aware REsidual Flow (SUREFlow)"*, a manipulation policy based on Mamba [12] that tackles long-horizon instability through action flow matching, efficient sequence modeling, and uncertainty-aware inference. The proposed framework formulates action generation as a continuous-time flow matching problem and learns a velocity field that maps noise to actions conditioned on visual observations, robot states, and task embeddings. To improve robustness during execution, it incorporates uncertainty-aware modeling that estimates prediction confidence alongside the action velocity field, as illustrated in Fig. 1. During inference, predicted uncertainty is used to selectively refine action sequences, mitigating error accumulation and enable

stable long-horizon rollout execution, while maintaining a lightweight multimodal backbone and consistently outperforming recent state-of-the-art (SOTA) robot manipulation baselines across multiple benchmarks. Our key contributions are as follows:

- We introduce an *Uncertainty-aware Residual Flow (URFlow)* refinement framework within conditional flow matching for generative robot manipulation. URFlow learns input-dependent velocity variance and selectively corrects unreliable action dimensions with negligible computational overhead, stabilizing iterative flow integration under extended rollouts.
- We proposed integrating a lightweight *Memory-Guided Action Decoder (MGAD)* that re-attends learnable action queries to multimodal memory representations, enabling context-aware and structured action refinement. This mechanism enhances temporal conditioning and improves robustness in generative action synthesis, as validated through ablation studies.
- To the best of our knowledge, integrating uncertainty-aware residual refinement within a state-space generative flow policy has not been previously explored. We address this gap by developing *SUREFlow*, a Mamba-based VLA architecture that unifies conditional flow matching with URFlow and MGAD. Leveraging linear-time sequence modeling, SUREFlow enables scalable and efficient generative control with only 179M parameters, avoiding reliance on large multimodal backbones.

II. Related Works

A. VLAs for Robot Manipulation

Recent advances in VLA models have significantly expanded the scope of generalist robotic manipulation by leveraging large-scale vision-language pretraining. Early foundation models such as RT-1 [13] and RT-2 [2] demonstrated that pretrained vision-language representations can be adapted for discretized robot action prediction, enabling zero-shot generalization across diverse tasks and embodiments. Building on this paradigm, OpenVLA scaled to 7B parameters [14], while Octo trained a transformer-based generalist policy across heterogeneous trajectories to promote multi-embodiment generalization [15]. Subsequent efforts focused on efficiency and architectural refinement. TinyVLA reduced model size while retaining competitive performance [16], and $\pi_0.5$ [17] improved deployment efficiency of pretrained VLA backbones [17]. QueST introduced structured tokenization and query-based conditioning [18], whereas TraceVLA injected spatial-temporal visual traces without modifying the backbone [19]. In parallel, Mamba [12] has been explored as an efficient alternative to transformer backbones. MaIL [3] leverages Mamba for long-horizon imitation learning, and RoboMamba [20] integrates Mamba within a VLA framework for multimodal policy learning.

Beyond autoregressive VLAs, generative policies have emerged as an alternative paradigm for action synthesis. Diffusion-based approaches such as Diffusion Policy [7] and 3D Diffusion Policy [21] model multimodal action distributions through iterative denoising, but require multiple sampling steps during inference. Flow-based methods, including Flow Matching [8] and continuous normalizing flows [22], instead learn velocity fields that transport a prior distribution toward expert actions without explicit diffusion. While promising for generative modeling, these formulations typically assume homoscedastic residuals and lack explicit uncertainty-aware refinement mechanisms for stabilizing extended-horizon rollout execution.

Despite these advances, many VLA models rely on large transformer backbones and autoregressive decoding, resulting in substantial computational overhead. Furthermore, explicit modeling of input-dependent uncertainty is typically absent, reducing robustness during iterative rollout and extended rollout execution. These limitations motivate exploring efficient state-space backbones [12] combined with uncertainty-aware generative refinement to improve scalability and extended-horizon execution stability.

B. Uncertainty Modeling and Residual Learning

Uncertainty estimation has been widely studied in deep learning and control. Kendall and Gal [23] introduced heteroscedastic regression for modeling input-dependent variance, while uncertainty-aware reinforcement learning (RL) improves exploration and safety [24], [25]. However, these approaches are generally applied to value estimation or supervised prediction rather than generative action transport. Additionally, residual learning has proven effective in stabilizing robotic control. Residual RL [26] combines model-based controllers with learned residual policies to correct systematic errors. Related strategies learn additive corrections to mitigate model mismatch, yet such residual formulations have not been integrated with flow-based generative policies.

III. Problem Formulation

We consider the task of learning a goal-conditioned robot manipulation policy $\pi(a \mid o, g)$ that maps multimodal observations to a sequence of continuous actions. Here, $o \in \mathcal{O}$ denotes the current observation, g is a natural language task instruction, and $a \in \mathcal{A}$ is the target action sequence. The observation consists of multi-view RGB images and robot proprioceptive states as follows:

$$o_t = \{I_t^{(1)}, I_t^{(2)}, s_t\}, \tag{1}$$

where $I_t^{(1)}$ and $I_t^{(2)}$ denote the agent's view and eye-in-hand RGB images, respectively, and s_t denotes the robot joint and gripper states. The task instruction g is encoded as a fixed task embedding e using a precomputed language encoder.

In our proposed SUREFlow, we formulate policy learning through the lens of conditional flow matching. Rather than directly regressing actions, the objective is to learn a time-dependent vector field v_t that induces a probability path $p_t(x)$ transforming a simple prior distribution p_1 (Gaussian noise) into the expert action distribution p_0. To construct this path, we define a conditional probability trajectory x_t for a continuous time variable $t \in [0, 1]$ as a linear interpolation

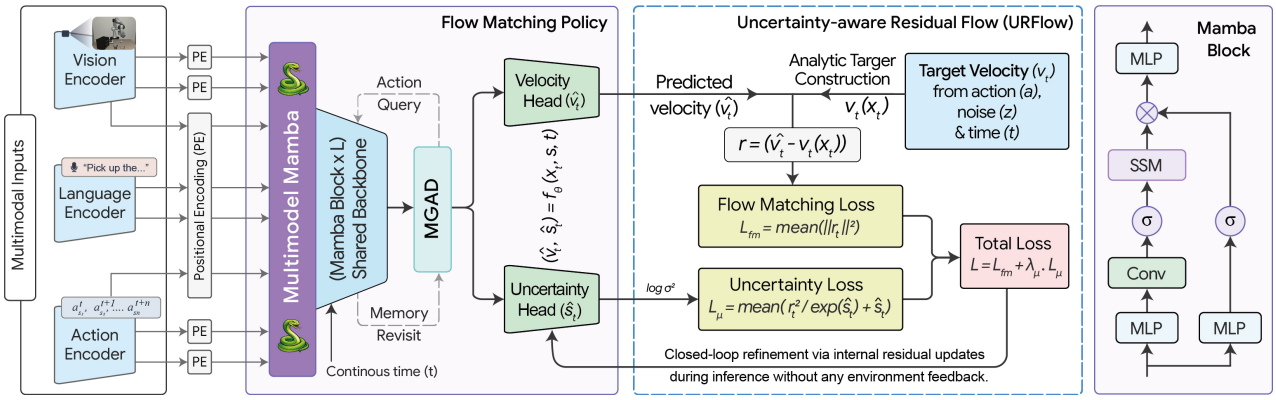

Fig. 2. Overall architecture of our proposed *SUREFlow*. Multimodal inputs are encoded and fused into a unified token representation. The fused tokens are processed by a shared backbone (stack of L number of Mamba blocks) conditioned on continuous flow time with the *'Memory-Guided Action Decoder (MGAD)'* module to enhance temporal conditioning and contextual reasoning. The flow-matching policy predicts an action velocity field and an uncertainty estimate, yielding our Uncertainty-aware Residual Flow (URFlow), which constructs flow-matching and uncertainty-weighted losses during training. At inference time, URFlow performs closed-loop refinement via internal residual updates guided by predicted uncertainty, improving robustness during long-horizon execution without online action correction or environment feedback.

between a ground truth action $a \sim p_0$ and a noise sample $z \sim \mathcal{N}(0, I)$ as follows:

$$x_t = (1-t)z + ta. \tag{2}$$

The analytic vector field that generates this transformation is given by the target velocity as $v_t(x_t) = z - a$. We model the learned conditional velocity field as $\hat{v}_t = f_\theta(x_t, s, t)$, where $s = \psi(o_t, e)$ denotes the fused multimodal conditioning representation. Unlike standard formulations, *SUREFlow* treats the discrepancy between the predicted and analytic target velocity as a residual flow matching problem. We hypothesize that this residual captures informative structure related to task complexity and data uncertainty, which can be exploited to refine policy behavior.

IV. Proposed Method: SUREFlow

We present SUREFlow, an end-to-end VLA framework that integrates uncertainty-aware residual flow matching within a unified state-space backbone. The overall architecture is illustrated in Fig. 2. Given multimodal inputs comprising RGB observations, robot proprioceptive states, and task language embeddings, each modality is encoded into a shared latent space through dedicated encoders and multimodal fusion. The resulting token sequence is processed by a Mamba-based state-space backbone, enabling efficient modeling of long-range temporal dependencies.

On top of this backbone, a flow matching head predicts a conditional velocity field that transports Gaussian noise toward expert action trajectories. In parallel, an auxiliary head estimates input-dependent uncertainty over the predicted residual. Unlike conventional diffusion or flow-based policies that optimize uniform velocity residuals, SUREFlow explicitly models heteroscedastic uncertainty, formulating action generation as a residual learning problem. This design supports training-time closed-loop refinement and enables uncertainty-guided closed-loop residual refinement during inference, minimizing the discrepancy between the predicted and analytic target velocities. To further enhance conditioning and expressiveness, the framework optionally incorporates Flow Conditioned Modulation (FCM) for timestep-dependent feature modulation and MGAD to ground learnable action queries in multimodal observation memory.

A. Encoding and Conditioning

The first stage of SUREFlow is responsible for transforming raw multimodal inputs into a unified latent representation suitable for conditional flow matching. At each time step t, the observation o_t contains two RGB images $I_t^{(1)}$ and $I_t^{(2)}$ corresponding to the agent view and the eye-in-hand view. Each image is processed independently using a convolutional visual encoder. Formally, for each camera view $k \in \{1, 2\}$, we compute the visual embedding as follows:

$$z_t^{(k)} = f_{\text{img}}^{(k)}(I_t^{(k)}) \in \mathbb{R}^d, \tag{3}$$

where $f_{\text{img}}^{(k)}$ denotes a ResNet-18 backbone with Group Normalization, and $z_t^{(k)}$ is the resulting visual embedding. This produces one visual token per camera view at each t.

The robot joint and gripper states s_t are embedded using a linear projection to obtain a proprioceptive token defined as $z_t^{(s)} = f_{\text{state}}(s_t)$, where f_{state} is a learnable affine mapping. This token captures the instantaneous physical configuration of the robot and complements the visual observations. The natural language instruction g is encoded using a pretrained language encoder to obtain a fixed task embedding e, which remains constant throughout the trajectory and serves as a global conditioning signal for the manipulation goal. To incorporate the continuous flow time variable $t \in [0, 1]$ introduced in Sec. III, we use a learnable time embedding $\phi(t) \in \mathbb{R}^d$. This embedding enables the policy to adapt its behavior along the probability path connecting the noise distribution p_1 and the expert action distribution p_0. The encoded components are combined to form a unified token sequence at time step t as follows:

$$X_t = \left[\phi(t),\ e,\ z_t^{(1)},\ z_t^{(2)},\ z_t^{(s)},\ x_t\right], \tag{4}$$

where x_t denotes the current action token along the probability path defined in Sec. III. This sequence aggregates all conditioning information required for action generation, including task context, perception, robot state, and flow time.

This unified token sequence X_t serves as the input to the sequence modeling component described in the following subsection, where temporal dependencies and conditional action generation are performed.

1) Flow Conditioned Modulation (FCM): Beyond explicit concatenation of the flow-time token, SUREFlow employs FCM to inject temporal conditioning directly into intermediate feature representations. FCM leverages the time embedding $\phi(t)$ to perform feature-wise affine modulation, enabling smooth conditioning of the sequence model without modifying the token structure. In practice, $\phi(t)$ is implemented using a sinusoidal frequency embedding followed by a two-layer MLP, producing a d-dimensional vector.

Let $U \in \mathbb{R}^{B\times N\times d}$ denote a sequence of N token features with embedding dimension d, where U may represent state tokens, action tokens, task tokens, or decoded action features. From the flow-time embedding $\phi(t)$, we extract a conditioning vector $c \in \mathbb{R}^{B\times d}$ (selecting the first token when represented as a sequence). Two learnable linear projections generate feature-wise scale and shift parameters as follows:

$$\gamma = W_\gamma c + b_\gamma, \qquad \beta = W_\beta c + b_\beta, \tag{5}$$

where $\gamma, \beta \in \mathbb{R}^{B\times d}$. The modulated representation is computed via broadcasting across the token dimension:

$$\mathrm{FCM}(U, \phi(t)) = U \odot (1+\gamma) + \beta, \tag{6}$$

where $\odot$ denotes element wise multiplication and γ, β are broadcast along the token dimension. To ensure stable optimization, all modulation parameters are initialized to zero, yielding $\gamma = 0$ and $\beta = 0$ at initialization, such that FCM initially acts as an identity mapping.

In SUREFlow, FCM is applied to embedded state tokens, embedded action tokens, and decoded action features prior to the final prediction head. When goal conditioning is enabled, it is also applied to the task embedding. This design introduces continuous flow-time awareness throughout the network while preserving architectural simplicity.

B. Sequence Modeling and Action Generation

Given the conditioned token sequence X_t constructed in Eq. (4), *SUREFlow* performs joint sequence modeling and conditional action generation using a shared Mamba [12] backbone. This component is responsible for integrating multimodal context, temporal structure, and flow time conditioning to predict the action velocity field required for conditional flow matching.

The token sequence X_t is processed by a stack of L layers, which serve as the shared sequence backbone of the policy. Each layer implements a selective State-space model (SSM) [12] that updates hidden states through linear recurrent dynamics combined with input-dependent gating. SSM enables efficient modeling of long-range temporal dependencies with linear computational complexity, making it well-suited for long-horizon execution and temporal modeling. If we formally let $H_t^{(0)} = X_t$ to denote the input token sequence, the sequence backbone is computed as follows:

$$H_t^{(\ell)} = \mathrm{Mamba}_\ell\big(H_t^{(\ell-1)}\big), \quad \ell = 1, \dots, L, \tag{7}$$

where $H_t^{(\ell)}$ denotes the hidden token representations at layer ℓ. The final representation $H_t^{(L)}$ encodes joint information from visual observations, robot states, task context, flow time, and the current action token.

1) Memory-Guided Action Decoder (MGAD): To further enhance the interaction between action representations and multimodal context, SUREFlow optionally employs MGAD that treats the action token as a learnable query and allows it to attend to the remaining tokens in $H_t^{(L)}$, including visual, proprioceptive, and task tokens. This mechanism enables the action representation to selectively aggregate relevant contextual information before velocity prediction.

Let $h_t^{(a)}$ denote the action token extracted from $H_t^{(L)}$. The MGAD module in SUREFlow updates this token as follows:

$$\tilde{h}_t^{(a)} = \mathrm{MGAD}\big(h_t^{(a)}, H_t^{(L)}\big), \tag{8}$$

where $\mathrm{MGAD}(\cdot)$ denotes a cross-attention module parameterized by the policy. Unlike full transformer decoders that employ multiple learnable action queries and stacked decoder layers, MGAD performs a lightweight, single-step cross-attention refinement over the action token, preserving architectural efficiency. When MGAD is not used, the original action token $h_t^{(a)}$ is passed directly to both the *policy head* and the *uncertainty head*.

2) Velocity and Uncertainty Prediction: The refined action representation is mapped to the parameters required for conditional flow matching. Specifically, from the decoded action features $\tilde{h}_t^{(a)}$, the policy predicts both the velocity and the associated uncertainty as follows:

$$\hat{v}_t = f_v\Big(\tilde{h}_t^{(a)}\Big), \qquad \hat{s}_t = f_s\Big(\tilde{h}_t^{(a)}\Big), \tag{9}$$

where f_v and f_s are linear projection heads with independent parameters. Together with the shared backbone defined in Eq. (7), they instantiate the overall policy $f_\theta(x_t, s, t)$, where the backbone produces the decoded action representation $\tilde{h}_t^{(a)}$ and the projection heads map it to velocity and uncertainty predictions. The predicted velocity $\hat{v}_t$ therefore corresponds to the learned conditional vector field $f_\theta(x_t, s, t)$, while $\hat{s}_t$ estimates the log-variance ($\log \sigma^2$) of the residual and is used for uncertainty-aware modeling and refinement.

C. Uncertainty-aware Residual Flow (URFlow)

Standard conditional flow matching learns a deterministic approximation of the analytic target velocity, implicitly treating all residual errors as equally reliable. During extended rollout execution, demonstrations can be noisy or ambiguous, and the reliability of the learned flow may vary across action dimensions and contexts. SUREFlow addresses this limitation by introducing an uncertainty-aware residual formulation within conditional flow matching, inspired by heteroscedastic regression for modeling input-dependent uncertainty [26] and residual learning strategies for stabilizing robot control [23]. URFlow explicitly models heteroscedastic uncertainty in the learned flow velocity field, enabling selective refinement of unreliable action dimensions and mitigating compounding integration errors during iterative

Algorithm 1 Pseudocode of URFlow in our SUREFlow.

Require: Action $a \sim p_0$, noise sample z, flow time t, observation o_t, instruction g, task embedding e, policy f_θ, uncertainty weight λ_u, threshold τ, refinement steps K, step size η.
Ensure: Training loss $\mathcal{L}$ and refined action sequence a.
1: Encoding: compute fused latent $s = \psi(o_t, e)$
2: Path construction: $x_t \leftarrow (1-t)z + ta$
3: Target velocity: $v_{\mathrm{t}}(x_t) \leftarrow z - a$
4: Predictions: $(\hat{v}_t, \hat{s}_t) \leftarrow f_\theta(x_t, s, t)$
5: Residual: $r_t \leftarrow \hat{v}_t - v_{\mathrm{t}}(x_t)$
6: Flow matching loss: $\mathbb{E}_{a,z,t}\left[\|r_t\|_2^2\right]$
7: Uncertainty loss: $\mathcal{L}_u \leftarrow \|r_t\|_2^2 / \exp(\hat{s}_t) + \hat{s}_t$
8: Total loss: $\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{FM}} + \lambda_u \mathcal{L}_u$
9: return $\mathcal{L}$ ▷ *Used during training*

Inference time uncertainty-aware action refinement
10: **for** $k = 1$ to K **do**
11: At $t = 0$: $(\hat{v}_0, \hat{s}_0) \leftarrow f_\theta(a, s, 0)$
12: $\mathcal{M} \leftarrow \{i \mid \hat{s}_{0,i} > \tau\}$ ▷ *Compute uncertainty mask*
13: **for all** $i \in \mathcal{M}$ **do**
14: $a_i \leftarrow a_i - \eta \hat{v}_{0,i}$
15: **end for**
16: **end for**
17: return refined a

flow execution. To the best of our knowledge, such an integration has not been explored for improving long-horizon rollout stability in generative robot control.

1) Residual Flow Definition: As defined in Sec. III, the probability path is x_t with $a \sim p_0$ and $z \sim \mathcal{N}(0, I)$, and the analytic target velocity is $v_{\mathrm{t}}(x_t)$. Specifically, our policy f_θ predicts both the velocity $\hat{v}_t$ and its uncertainty $\hat{s}_t$ via a function that is shown in line 4 of Alg. 1. This is instantiated via a shared backbone defined in Eq. (7), followed by the projection heads f_v and f_s defined in Eq. (9). Given the predicted velocity $\hat{v}_t$ produced by the policy f_θ, we define the residual flow as r_t, as in line 5 of Alg. 1.

2) Training Objective: A standard flow matching loss is then optimized together with an uncertainty-aware residual loss $\mathcal{L}_{\mathrm{u}}$ as in Alg. 1 (line 7). The uncertainty-aware residual loss models r_t as heteroscedastic and penalizes residuals under the predicted variance. The final loss training objective is defined in Alg. 1 (line 8), where λ_u controls the contribution of the uncertainty term.

3) Inference Time Refinement: At inference time, the policy generates an action sequence by integrating the learned conditional velocity field $\hat{v}_t$ predicted by the policy to transport an initial noise sample toward the action distribution. To enhance stability in extended rollout steps, URFlow performs a deterministic refinement procedure guided by the predicted uncertainty $\hat{s}_t$ introduced in Eq. (9). Given a generated action sequence a, we evaluate the predicted uncertainty at $t = 0$ and construct a mask over unreliable action dimensions as defined in line 12 of Alg. 1, where τ is a predefined uncertainty threshold and $\hat{s}_{0,i}$ denotes the i-th dimension of the predicted log-variance at $t = 0$. For dimensions $i \in \mathcal{M}$, the action is refined using the predicted velocity at $t = 0$ via $a_i \leftarrow a_i - \eta \hat{v}_{0,i}$, where η is a fixed step size. This update is applied for a small number of refinement steps, and it operates entirely within the model without any online environment feedback or action correction.

V. Experimental Results

A. Experimental Setup

1) Implementation Details: All models are implemented in `PyTorch` and trained with AdamW using a learning rate of 1×10^{-4}, momentum 0.9, and weight decay 0.05. Training is conducted for 200 epochs with a batch size of 256 on NVIDIA RTX 4090 GPUs. For flow matching, the time variable $t \in [0, 1]$ is uniformly sampled per batch, with noise $z \sim \mathcal{N}(0, I)$ used to construct x_t as in Eq. (2). During inference, we use 50 flow integration steps. The time variable is provided as `sigma` and embedded via `TimeEmbedding`, corresponding to $\phi(t)$. The MGAD module uses 10 learnable action queries with a single-layer 4-head self and cross-attention decoder (head dim 64), followed by a 2-layer FFN (hidden 512) with LayerNorm. The output is linearly projected from 256 to 7 action dimensions. For perception, we use ResNet-18 visual encoders (one per RGB view) with 256-dimensional latent projections, and a CLIP ViT-B/32 text encoder for language conditioning. Actions are encoded from $\mathbb{R}^7$ to the shared 256-dimensional latent space using a linear embedding layer. The Mamba backbone consists of 5 layers with hidden dimension 256 and intermediate dimension 256. All token embeddings, including visual, state, task, and action tokens, share a common embedding dimension of 256. URFlow is enabled with $\lambda_u = 0.001$ without overpowering the primary flow-matching objective. During inference, refinement is triggered only when the predicted uncertainty exceeds $\tau = 0.9$, and each selected action is refined for 3 iterations. For ablation studies, Success Rate (SR) is averaged across all evaluation tasks, where each task is executed for 50 rollouts and each rollout executed for 350 steps to ensure stable performance estimation.

2) Dataset & Models: We evaluate SUREFlow on Meta-World [27], LIBERO [28], and LIBERO-PRO [29] to assess multi-task generalization, compositional reasoning, and long-horizon robustness. We compare SUREFlow with representative VLA and generative visuomotor policies spanning transformer, diffusion, and state-space architectures, including large-scale VLAs such as OpenVLA [14], Octo [15], and $\pi_{0.5}$ [17]. All models follow their official training and evaluation protocols for fair comparison. Overall, our experiments aim to address the following key Research Questions (RQs):

- RQ1: Can URFlow reduce compounding integration errors in generative flow-based control?
- RQ2: Does URFlow matching improve stability and success under extended rollout horizons?
- RQ3: Can URFlow achieve long-horizon robustness comparable to large-scale VLA models, without relying on billion-parameter backbones?
- RQ4: How does uncertainty calibration interact with selective refinement?

B. Comparison with SOTA Methods

To address RQ1 and RQ2, we first evaluate SUREFlow on the LIBERO and Meta-World benchmarks. LIBERO assesses multi-task generalization across spatial, object, goal,

TABLE I

PERFORMANCE COMPARISON ON LIBERO AND META-WORLD BENCHMARKS. ALL VALUES DENOTE SUCCESS RATE (%). THE HIGHEST SUCCESS RATES ARE INDICATED IN BOLD, AND THE SECOND-HIGHEST VALUES ARE UNDERLINED.

LIBERO							Meta-World						
Method	Venue	Spatial	Object	Goal	Long	Average	Method	Venue	Easy	Medium	Hard	Very Hard	Average
Octo [15]	RSS'24	78.9	85.7	84.6	51.1	75.1	DP3 [21]	RSS'24	90.9	61.6	31.7	49.0	74.4
QueST [18]	NeurIPS'24	<u>89.0</u>	<u>90.0</u>	<u>88.4</u>	<u>87.0</u>	<u>88.6</u>	Diffusion Policy [7]	IJRR'25	83.6	31.1	9.0	26.6	37.6
MaIL [3]	CoRL'24	53.8	81.5	56.3	41.7	58.3	Mamba Policy [1]	IROS'25	<u>95.4</u>	<u>92.2</u>	<u>48.3</u>	<u>71.2</u>	<u>76.8</u>
TraceVLA [19]	ICLR'25	84.6	85.2	75.1	54.1	74.8	TinyVLA-H [16]	ICRA'25	77.6	21.5	11.4	15.8	31.6
SUREFlow	**(Ours)**	**94.0**	**94.4**	**92.5**	**89.5**	**92.60**	**SUREFlow**	**(Ours)**	**97.8**	**93.5**	**86.1**	**75.9**	**88.32**

TABLE II

LIBERO-PRO MODEL LEADERBOARD SHOWING NORMALIZED SUCCESS RATES UNDER FIVE PERTURBATION TYPES ACROSS FOUR BENCHMARKS.

Model	LIBERO Goal					LIBERO Spatial					LIBERO 10					LIBERO Object					Average	Params ↓
	Obj	Pos	Sem	Task	Env	Obj	Pos	Sem	Task	Env	Obj	Pos	Sem	Task	Env	Obj	Pos	Sem	Task	Env	SR ↑	
OpenVLA [14]	0.96	0.00	0.98	0.00	0.98	0.97	0.00	0.97	0.00	0.89	0.81	0.00	0.96	0.00	0.85	0.98	0.00	0.98	0.00	0.00	<u>0.52</u>	7B
π_0 [30]	0.94	0.00	0.93	0.00	0.39	0.95	0.00	0.97	0.00	0.60	0.79	0.00	0.82	0.00	0.27	0.94	0.00	0.90	0.00	0.29	0.44	3B
$\pi_{0.5}$ [17]	0.97	0.38	0.97	0.00	0.46	0.97	0.20	0.97	0.01	0.46	0.92	0.08	0.93	0.01	0.46	0.98	0.17	0.96	0.01	0.73	**0.53**	3B
SUREFlow (Ours)	0.91	0.00	0.95	0.00	0.92	0.90	0.00	0.89	0.00	0.90	0.34	0.00	0.83	0.00	0.72	0.70	0.00	0.94	0.10	0.94	0.50	**179.1M**

and long-horizon suites. As summarized in Table I, SUREFlow achieves an average SR of 92.60%, outperforming transformer-based models like Octo by 17.5% and SSM-based methods MaIL by 34.3%. Representative simulation results across the LIBERO suites are presented in Fig. 3. On the Meta-World, which emphasizes diverse manipulation skills with varying difficulty levels, SUREFlow achieves an average SR of 88.32%, surpassing DP3 at 74.4%, Diffusion Policy at 37.6%, Mamba Policy at 76.8%, and TinyVLA H at 31.6%. Notably, compared to Diffusion Policy, SUREFlow improves average success by nearly 78%, highlighting the advantage of flow matching over iterative denoising for efficient and stable action generation.

Moreover, 34.3% gain over MaIL and 11.52% improvement over Mamba Policy, which are both based on Mamba backbone, further confirms that the performance gains are not solely attributable to the State-space backbone, but arise from our proposed URFlow-based uncertainty-aware residual modeling and closed-loop refinement. Thus, integrating URFlow within a state-space backbone improves long-horizon stability and robustness, directly addressing RQ1.

Unlike autoregressive transformer VLAs that depend on large-scale pretrained backbones, SUREFlow learns a conditional velocity field with explicit uncertainty calibration, enabling selective refinement of unreliable action dimensions during rollout. The strong performance across easy to very hard task categories, together with improvements under extended horizons, directly supports RQ2 by demonstrating that uncertainty-guided refinement enhances stability under increasing task complexity and rollout length.

C. Robustness Under Distribution Shift and Efficiency

To evaluate robustness under distribution shift and address RQ3, we compare SUREFlow with large-scale VLAs on LIBERO-PRO, which measures normalized SR under object position, semantic, task, and environment perturbations. Our SUREFlow achieves an average SR of 0.50 using 179.1M parameters. While OpenVLA (7B) and $\pi_{0.5}$ (3B) obtain 0.52 and 0.53, respectively, SUREFlow remains competitive despite being more than an order of magnitude smaller. It also outperforms π_0 (3B), which achieves an SR of 0.44.

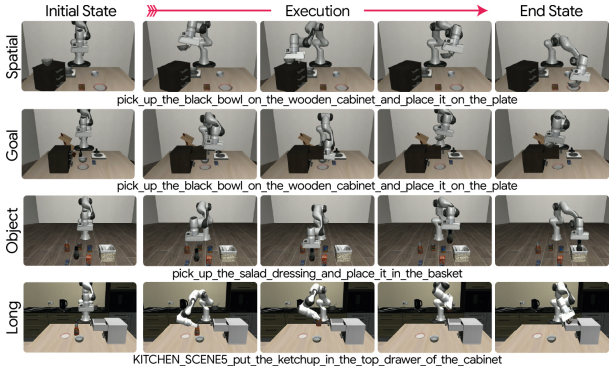

Fig. 3. Example evaluation outcome of our SUREFlow on LIBERO.

Despite operating with only 179.1M parameters, SUREFlow attains performance close to OpenVLA (7B) and $\pi_{0.5}$ (3B) under diverse perturbation settings, indicating that URFlow effectively compensates for model scale. The Mamba backbone in our SUREFlow enables linear-time modeling without quadratic transformer cost, while uncertainty-guided refinement stabilizes extended-horizon rollouts and reduces compounding errors. These results directly address RQ3 by demonstrating that our model SUREFlow achieves long-horizon robustness comparable to large VLA models.

1) Stability and Uncertainty Calibration Analysis: To further address RQ2, RQ3, and RQ4, we analyze long-horizon rollout behavior and uncertainty dynamics in Fig. 4. An extended rollout is particularly prone to compounding errors during open-loop execution, where small velocity inaccuracies accumulate over time. Fig. 4(a) shows that with URFlow enabled, performance improves steadily and remains stable as the rollout horizon increases. In contrast, without URFlow, the success rate decreases more rapidly and becomes less stable as the number of steps grows. This directly supports RQ2 by demonstrating that URFlow matching improves stability and success under extended rollout horizons. Furthermore, the improved stability under increasing horizons achieved without scaling model size further supports RQ3. Because URFlow achieves long-horizon rollout robustness through calibrated residual refinement rather than reliance on billion-parameter backbones. Fig. 4(b) shows refinement activation rates under variable uncertainty thresholds τ. Lower τ val-

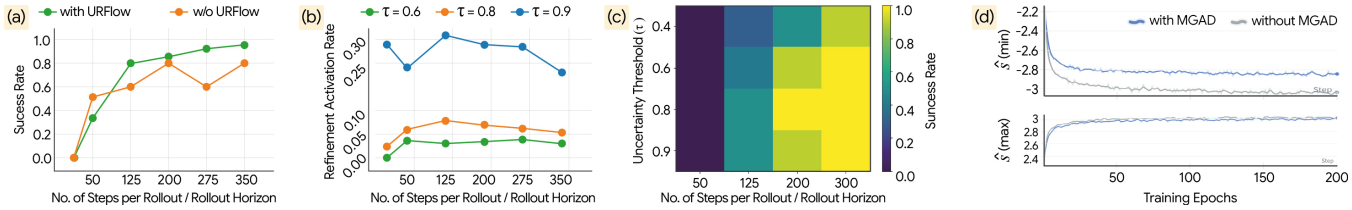

Fig. 4. Long-horizon rollout stability and URFlow-based refinement analysis on LIBERO Long. (a) Refinement activation rate under different uncertainty thresholds $\tau \in \{0.6, 0.8, 0.9\}$ as rollout horizon increases; lower τ triggers more frequent updates, while higher τ yields more selective refinement. (b) Success rate vs. rollout horizon comparing SUREFlow with and without URFlow, showing slower performance degradation under extended rollouts when uncertainty-guided refinement is enabled. (c) Heatmap of success across thresholds and horizons, illustrating stable performance for moderate τ under long-horizon settings. (d) Evolution of predicted uncertainty statistics during training, indicating the stability with and without our MGAD module.

TABLE III

ABLATION STUDY ON SUREFLOW COMPONENTS.

Variant	FCM	MGAD	URFlow	SR↑	Params (M)↓	FLOPs (G)↓
+ FCM	✓	×	×	0.72	178.30	6.48
+ MGAD	×	✓	×	0.77	178.57	6.78
+ URFlow	×	×	✓	0.81	177.78	6.41
+ FCM + MGAD	✓	✓	×	0.85	179.10	6.85
+ MGAD + URFlow	×	✓	✓	0.91	178.57	6.78
+ FCM + URFlow	✓	×	✓	0.87	178.30	6.48
SUREFlow (Baseline)	✓	✓	✓	0.92	179.10	6.85

TABLE IV

SENSITIVITY ANALYSIS OF URFLOW ON LIBERO (AVERAGE SR).

Effect of	Configuration	SR↑	Effect of	Configuration	SR↑
Uncertainty weight λ_u	$\lambda_u = 0.001$	**0.92**	Refinement steps K	$K = 2$	0.85
	$\lambda_u = 0.01$	0.87		$K = 3$	0.92
	$\lambda_u = 0.05$	0.79		$K = 5$	0.88

ues trigger more frequent residual updates, whereas higher values yield more selective refinement. The corresponding success heatmap in Fig. 4(c) reveals that moderate thresholds maintain stable performance across longer horizons, while overly aggressive or overly conservative refinement leads to reduced gains. These results address RQ4 by showing that uncertainty calibration directly governs the trade-off between refinement frequency and stability, enabling selective correction of unreliable action dimensions without introducing excessive updates.

2) Effect of MGAD on Uncertainty Calibration: Fig. 4(d) shows the evolution of predicted uncertainty statistics during training, comparing models with and without MGAD. The trends of $\hat{s}_{\min}$ and $\hat{s}_{\max}$ indicate that MGAD leads to faster convergence and a more stable uncertainty range. In contrast, removing MGAD results in consistently lower $\hat{s}_{\min}$ values, suggesting overconfident predictions. By moderating extreme values and maintaining a bounded uncertainty distribution, MGAD improves calibration and contributes to more stable behavior in extended rollout.

D. Ablation Study

1) Component Analysis: Table III evaluates the impact of FCM, MGAD, and URFlow. Individually, FCM achieves 0.72 SR, MGAD 0.77, and URFlow 0.81, with URFlow providing the greatest single-module improvement. Combining FCM and MGAD increases SR to 0.85, while MGAD with URFlow reaches 0.91, indicating strong complementary effects. The full SUREFlow model achieves the highest SR of 0.92. Notably, these improvements are obtained with only modest changes in parameter count and FLOPs, confirming that performance gains arise from improved modeling rather than brute-force scaling.

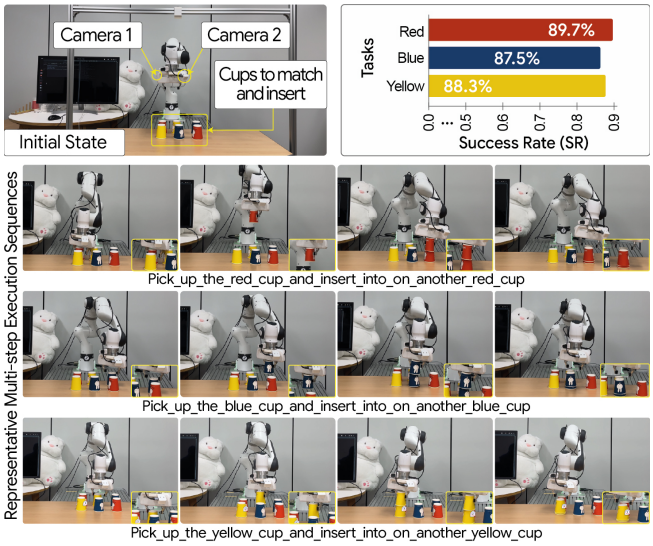

Fig. 5. Real-robot color-matching & cup insertion task by SUREFlow. Top: two-camera setup and SR over 20 rollouts per color. Bottom: representative multi-step execution sequences for red, blue, and yellow cups.

2) Sensitivity Analysis of URFlow: It is observed from the results in Table IV that moderate weighting improves stability and success. The best result of 0.92 is achieved with a properly calibrated λ_u, whereas excessive weighting degrades performance, emphasizing the need for balanced uncertainty regularization. Similarly, refinement depth influences robustness: 2 refinements step achieves 0.85 with moderate variance, while 3 steps attain 0.92 with improved stability; additional steps provide no further gains. These results confirm that properly calibrated uncertainty weighting and selective refinement are critical for mitigating compounding errors without overcorrection.

VI. REAL ROBOT EXPERIMENT

We evaluate SUREFlow on a real-world tabletop manipulation task using a Franka Emika arm with a parallel gripper and an Intel RealSense D405 RGB D camera, with inference running on an NVIDIA RTX 4090 GPU. The setup includes 6 cups in 3 colors, red, yellow, and blue, with two cups per color. Given a visual observation, the robot is required to (1) select one cup, (2) identify the second cup of the same color, and (3) insert the grasped cup onto its color-matched counterpart. This setup requires reliable color discrimination, precise grasping, and accurate spatial

alignment during insertion. A total of 20 independent rollouts were conducted across the three color pairs.

SUREFlow achieved an average SR of 88.5% over 20 rollouts per color (Fig. 5), consistently completing the color-matching pick-and-insert task. The model demonstrates reliable color-conditioned reasoning, stable grasping, and accurate insertion alignment. These results suggest that uncertainty-aware residual refinement enhances robustness under real-world perception noise and minor pose variations, supporting the practical effectiveness of SUREFlow.

VII. Conclusion

We presented SUREFlow, a state-space uncertainty-aware residual flow matching framework for robust generative robot manipulation. By integrating URFlow with a Mamba-based backbone and MGAD, our method models input-dependent uncertainty and performs selective residual refinement to mitigate compounding errors during extended rollouts. Extensive evaluations on LIBERO, Meta-World, and LIBERO-PRO demonstrate strong performance gains over both transformer-based and state-space baselines, while maintaining a lightweight design. Real-world experiments further validate practical robustness. These results highlight uncertainty-calibrated flow refinement as an effective and scalable direction for stable long-horizon robot control.

References

[1] J. Cao, Q. Zhang, J. Sun, J. Wang, H. Cheng, Y. Li, J. Ma, K. Wu, Z. Xu, Y. Shao, *et al.*, "Mamba policy: Towards efficient 3d diffusion policy with hybrid selective state models," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11359–11366, IEEE, 2025.

[2] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*, pp. 2165–2183, PMLR, 2023.

[3] X. Jia, Q. Wang, A. Donat, B. Xing, G. Li, H. Zhou, O. Celik, D. Blessing, R. Lioutikov, and G. Neumann, "Mail: Improving imitation learning with selective state space models," in *8th Annual Conference on Robot Learning*, 2024.

[4] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conference on robot learning*, pp. 158–168, PMLR, 2022.

[5] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," *Advances in neural information processing systems*, vol. 35, pp. 22955–22968, 2022.

[6] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *Conference on robot learning*, pp. 1113–1132, Pmlr, 2020.

[7] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, vol. 44, no. 10-11, pp. 1684–1704, 2025.

[8] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *The Eleventh International Conference on Learning Representations*, 2023.

[9] X. Ma, S. Patidar, I. Haughton, and S. James, "Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18081–18090, 2024.

[10] R. Liu, P. Zhou, Q. Luo, L. Sun, J. Cen, Y. Song, and Y. Yang, "Himacon: Discovering hierarchical manipulation concepts from unlabeled multi-modal data," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[11] T. Haarnoja, K. Hartikainen, P. Abbeel, and S. Levine, "Latent space policies for hierarchical reinforcement learning," in *International Conference on Machine Learning*, pp. 1851–1860, PMLR, 2018.

[12] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," in *First conference on language modeling*, 2024.

[13] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[14] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, "OpenVLA: An open-source vision-language-action model," in *8th Annual Conference on Robot Learning*, 2024.

[15] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine, "Octo: An open-source generalist robot policy," in *Proceedings of Robotics: Science and Systems*, (Delft, Netherlands), 2024.

[16] J. Wen, Y. Zhu, J. Li, M. Zhu, K. Wu, Z. Xu, N. Liu, R. Cheng, C. Shen, Y. Peng, *et al.*, "Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation," in *IEEE Robotics and Automation Letters (RA-L)*, 2025.

[17] K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. R. Equi, C. Finn, N. Fusai, and et al., "$\pi_{0.5}$: a vision-language-action model with open-world generalization," in *9th Annual Conference on Robot Learning*, 2025.

[18] A. Mete, H. Xue, A. Wilcox, Y. Chen, and A. Garg, "Quest: Self-supervised skill abstractions for learning continuous control," *Advances in Neural Information Processing Systems*, vol. 37, pp. 4062–4089, 2024.

[19] R. Zheng, Y. Liang, S. Huang, J. Gao, H. D. III, A. Kolobov, F. Huang, and J. Yang, "TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies," in *The Thirteenth International Conference on Learning Representations*, 2025.

[20] J. Liu, M. Liu, Z. Wang, P. An, X. Li, K. Zhou, S. Yang, R. Zhang, Y. Guo, and S. Zhang, "Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation," *Advances in Neural Information Processing Systems*, vol. 37, pp. 40085–40110, 2024.

[21] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," in *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[22] M. S. Albergo and E. Vanden-Eijnden, "Building normalizing flows with stochastic interpolants," in *The Eleventh International Conference on Learning Representations*, 2023.

[23] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," *Advances in neural information processing systems*, vol. 30, 2017.

[24] G. Liu, Y. Luo, O. Schulte, and P. Poupart, "Uncertainty-aware reinforcement learning for risk-sensitive player evaluation in sports game," *Advances in Neural Information Processing Systems*, vol. 35, pp. 20218–20231, 2022.

[25] S. Depeweg, J.-M. Hernandez-Lobato, F. Doshi-Velez, and S. Udluft, "Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning," in *International conference on machine learning*, pp. 1184–1193, PMLR, 2018.

[26] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 international conference on robotics and automation (ICRA)*, pp. 6023–6029, IEEE, 2019.

[27] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on robot learning*, pp. 1094–1100, PMLR, 2020.

[28] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "Libero: Benchmarking knowledge transfer for lifelong robot learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 44776–44791, 2023.

[29] X. Zhou, Y. Xu, G. Tie, Y. Chen, G. Zhang, D. Chu, P. Zhou, and L. Sun, "Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization," *arXiv preprint arXiv:2510.03827*, 2025.

[30] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, *et al.*, "*pi*_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.