2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

KAN We Flow? Advancing Robotic Manipulation with 3D Flow Matching via KAN & RWKV

Zhihao Chen[1], Yiyuan Ge[2,✉], Ziyang Wang[3]

***Abstract*— Diffusion-based visuomotor policies excel at modeling action distributions but are inference-inefficient, since recursively denoising from noise to policy requires many steps and heavy UNet backbones, which hinders deployment on resource-constrained robots. Flow matching alleviates the sampling burden by learning a one-step vector field, yet prior implementations still inherit large UNet-style architectures. In this work, we present KAN-We-Flow, a flow-matching policy that draws on recent advances in Receptance Weighted Key Value (RWKV) and Kolmogorov-Arnold Networks (KAN) from vision to build a lightweight and highly expressive backbone for 3D manipulation. Concretely, we introduce an RWKV-KAN block: an RWKV first performs efficient time/channel mixing to propagate task context, and a subsequent GroupKAN layer applies learnable spline-based, groupwise functional mappings to perform feature-wise nonlinear calibration of the action mapping on RWKV outputs. Moreover, we introduce an Action Consistency Regularization (ACR), a lightweight auxiliary loss that enforces alignment between predicted action trajectories and expert demonstrations via Euler extrapolation, providing additional supervision to stabilize training and improve policy precision. Without resorting to large UNets, our design reduces parameters by 86.8%, maintains fast runtime, and achieves state-of-the-art success rates on Adroit, Meta-World, and DexArt benchmarks. Our project page can be viewed in link .**

I. INTRODUCTION

Imitation learning [1]–[5] enables robots to acquire and reproduce specific skills by observing expert demonstrations, allowing them to execute delicate, highly complex behaviors that are difficult to specify with conventional programming. Among current approaches, generative-model-based policies [6]–[18] have become dominant, and they transfer advances from text-to-image generation to robotic imitation learning, achieving strong performance.

These generative methods can be broadly classified into two types: diffusion-based approaches [6]–[10] and flow matching paradigms [11]–[15]. Diffusion-based approaches model multi-modal action distributions by iterative denoising, which has proven effective in robotic manipulation.

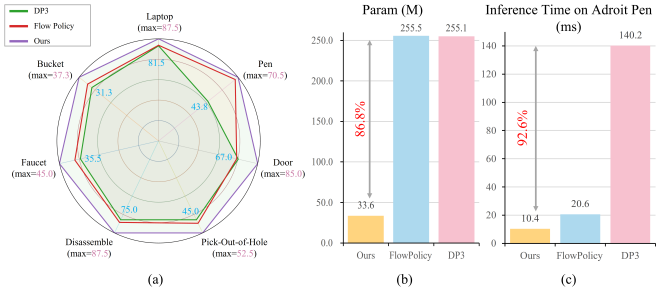

Fig. 1: Comparison of KAN-We-Flow with the state-of-the-art methods FlowPolicy and DP3 regarding accuracy, parameter, and inference time. (a) KAN-We-Flow achieves superior success rates among different benchmarks' tough tasks; (b) Our approach obtains an 86.8% parameter reduction, compared with FlowPolicy and DP3; (c) Compared with DP3, our KAN-We-Flow achieves 92.6% inference time decrease in the Adroit–Pen task, enabling real-time control.

However, a key limitation of diffusion policies is slow inference speed, producing each action requires many sampling steps, leading to high latency, unsuitable for real-time control [11]–[13]. To address this issue, flow matching methods are introduced, which directly learn a continuous vector field that transports noise to target actions, often enabling one-step generation of actions [14], [15]. Despite the strong success rates of both diffusion and flow-matching policies (Fig. 3 a), practical deployment remains bottlenecked by heavy UNet-style backbones that inflate parameters (Fig. 3 b). In diffusion, DP3 [7] is a representative case whose UNet exceeds ∼255 M parameters, and even a simplified DP3 speeds up mainly by trimming the UNet, underscoring the backbone as the dominant compute sink. For flow matching, they alleviate sampling inefficiency by directly learning a vector field, which often enables one-step action generation. It also delivers a runtime per step faster than diffusion baselines; for example, FlowPolicy [11] runs around 20 ms in the Adroit Pen task and is about 7 times faster than DP3 [7]. However, its flow implementations still inherit large UNet-like backbones (see Fig. 3 b), which limit edge deployment. To make visuomotor policies practical on resource-constrained robots, we thus seek an architecture that preserves the accuracy and inference speed of flow models while substantially reducing compute and sustaining long-horizon prediction, which are crucial for robot learning.

Recently, two efficient modeling advances, Receptance Weighted Key Value (RWKV) and Kolmogorov–Arnold

[1] Zhihao Chen is with the School of Intelligent Engineering and Automation, Beijing University of Posts and Telecommunications (BUPT), China, and also with Beijing Hydrogen Intelligence Technology Co. Ltd., China. zhihaochen666@bupt.edu.cn

[2,✉] Yiyuan Ge (corresponding author) is with the School of Electronic and Information Engineering, South China University of Technology, China. 202510182871@mail.scut.edu.cn

[3] Ziyang Wang with the Department of Computer Science at the University of Oxford, UK. ziyangwang@ieee.org

This work was supported by the Young Scientists Fund of NSFC (Grant No. 62406035), and Start-up Funding of Beijing University of Posts and Telecommunications (grant No.510224072).

Networks (KAN), offer an attractive path forward. RWKV attains transformer-level sequence modeling with linear complexity and has shown strong results across visual tasks [19]–[22]. KAN, grounded in the Kolmogorov–Arnold representation theorem, replaces heavy matrix-based feature mixing with stacks of learnable 1D nonlinearities (spline-like activations), enabling efficient approximation of multivariate functions with far fewer parameters [23]–[25]. Although these designs have demonstrated compelling efficiency in vision (e.g., Vision-RWKV [20] and U-KAN [23]), their application in the robotic domain remains largely unexplored, which motivates us to explore their potential to achieve efficient manipulation.

In this paper, we aim to propose an efficient RWKV and KAN-based policy that reduces the large number of parameters in previous methods and maintains the FlowPolicy's fast inference speed. We achieve this by introducing an RWKV and a GroupKAN layer, called the RWKV-KAN block. Specifically, RWKV's recurrent sequence modeling absorbs temporal correlations across the action trajectory, while the GroupKAN layer compactly represents the intricate visuomotor mapping. By marrying RWKV's sequential reasoning with KAN's compact expressiveness, our policy network achieves drastically reduced model size and faster inference without sacrificing precision. To further stabilize training without extra inference steps, we introduce an Action Consistency Regularization (ACR), a lightweight auxiliary objective that aligns the policy's predicted action trajectory with expert demonstrations via a simple Euler extrapolation from intermediate states to the end of the horizon. This adds a global end-point anchor to the flow-matching objective, reducing drift and exposure-bias effects while incurring negligible compute and no extra inference steps. Finally, we evaluate our approach across the Adroit, Meta-World, and DexArt benchmarks, and show that the KAN-We-Flow policy achieves state-of-the-art success rates in vision-based imitation learning. Our contributions are as follows:

- We introduce a novel flow-matching policy that integrates RWKV and GroupKAN modules. This is the first use of these architectures for robotic visuomotor policy learning, enabling faster inference speed with fewer parameters compared to previous methods.
- We propose the Action Consistency Regularization (ACR) that anchors predicted actions to demonstrations at the horizon, improving stability and precision at negligible cost.
- We conduct extensive experiments on Adroit, Meta-World, and DexArt benchmarks, demonstrating that our KAN-We-Flow policy achieves higher success rates than state-of-the-art baselines.

II. Related Work

A. Diffusion vs. Flow-Based Policies

Generative visuomotor policies for robotic manipulation mainly follow two paradigms: diffusion-based [7]–[9], [26]–[30] and flow-matching–based [11], [12], [15], [31], [32]. Diffusion policies model actions via conditional denoising, which is expressive and often stable to train, but typically require many iterative sampling steps and heavy backbones, leading to high latency and limited edge deployability (e.g., large UNet-style models in DP3 [7]). Recent variants reduce backbone or improve long-horizon control (e.g., state-space/transformer backbones [8], [26] and hierarchical designs [27]), yet the iterative sampling bottleneck remains.

Flow-matching methods instead generate actions in one or a few steps, enabling real-time inference and competitive success via temporal/frequency constraints and tailored regularization [12], [31], [32]. However, many implementations still depend on large UNet-like backbones or multi-stage pipelines (e.g., engineered refinement loops [32] or VLM-in-the-loop planning [15]), which can reintroduce compute overhead.

Overall, diffusion offers strong expressivity but is inference-inefficient, while flow matching improves inference efficiency yet often inherits heavyweight backbones. To address this, we propose a lightweight flow-matching policy that integrates RWKV and GroupKAN modules, achieving faster inference with fewer parameters than prior generative policies.

B. Current Policy Network Architectures

Modern visuomotor policy backbones can be categorised into three principal paradigms: (i) convolutional neural networks [7], [11], [12], (ii) transformers [26], [33], [34], and (iii) emerging state-space (Mamba) models [8], [35].

For the CNN paradigm, DP3 [7] marries compact 3D point-cloud encodings with a diffusion UNet to denoise noise into action sequences, delivering strong generalization across spatial/appearance shifts while remaining diffusion-style in inference. FlowPolicy [11] keeps the CNN backbone but replaces iterative denoising with consistency flow matching so that a single network evaluation yields actions with $7\times$ faster inference yet competitive success. MeanFlow [12] also uses point-cloud inputs with a lightweight CNN stack, learning interval-averaged velocities to achieve genuine 1-NFE trajectory generation and millisecond-level latency while outperforming DP3 and FlowPolicy.

For transformer-based designs, DiT-Block [33] Policy builds an encoder–decoder diffusion Transformer with adaLN-Zero stabilization, scaling to long-horizon dexterous tasks and multi-modal conditioning. Diffusion Transformer Policy [26] directly uses a large causal Transformer to denoise continuous action chunks from images/language, achieving strong generalist performance but still requiring multiple denoising steps at test time. MTDP [34] introduces a Modulated Attention decoder that better fuses guiding conditions and, with a DDIM variant, reduces sampling steps while maintaining performance.

Finally, for state-space (Mamba) backbones, MambaPolicy [8] replaces heavy UNets with selective state-space modules, reporting $>80\%$ parameter reductions and notable FLOPs savings without sacrificing success on Adroit, Meta-World, and DexArt.

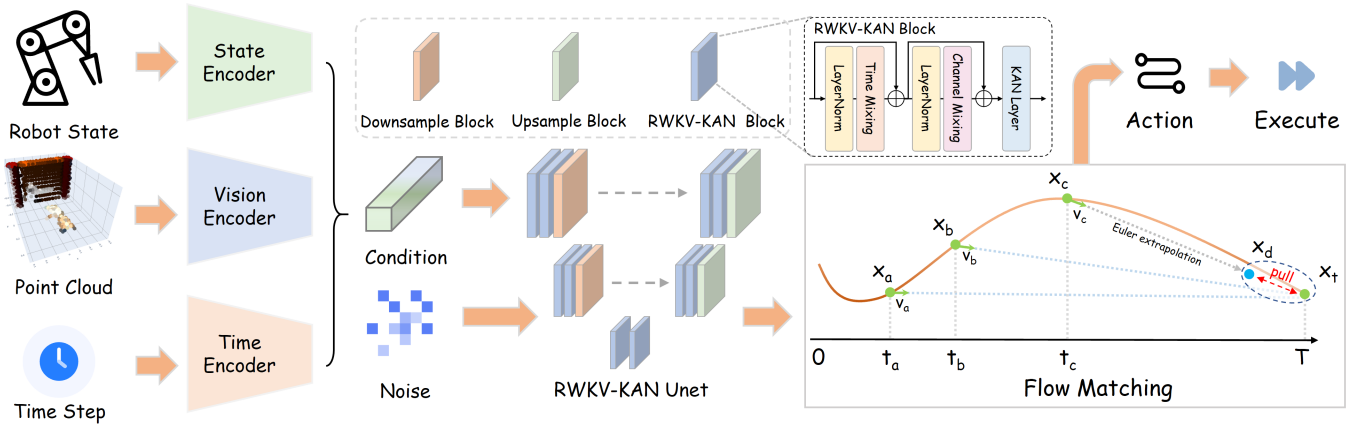

Fig. 2: **Overview of KAN-We-Flow.** The policy receives a noised action and a condition that comprises three encoded parts, a point-cloud perception embedding, a robot-state embedding, and a time embedding. The concatenated representation is processed by a lightweight RWKV-KAN U-shaped backbone instead of a large UNet-style backbone; RWKV mixes long-range time/channel context with linear complexity, while KAN performs learnable spline-based feature calibration. Then, a straight-line flow is learned with conditional consistency flow matching to produce a one-step velocity field, and the resulting actions are generated at real-time inference speed; an additional action consistency regularization aligns Euler-extrapolated trajectories with demonstrations to stabilize training.

III. Method

A. Overview

As shown in Fig. 2, we first encode a single-view point cloud and robot state into a compact condition. This condition is then fed into a lightweight RWKV-KAN UNet, which predicts a one-step velocity field. RWKV provides linear-time temporal mixing to propagate long-horizon context, while GroupKAN performs parameter-efficient per-channel functional calibration. Next, we adopt conditional consistency flow matching (CFM) for single-step action generation, and introduce an Action Consistency Regularization (ACR) term. The ACR anchors the one-step decoded action at the horizon ($t \to 1$) to the expert demonstration via Euler extrapolation, which provides additional supervision to stabilize training without extra inference steps.

B. RWKV-KAN UNet

The RWKV-KAN UNet is a three-stage encoder–decoder where each stage stacks RWKV-KAN blocks: RWKV mixes long-range temporal context with linear complexity, while GroupKAN performs per-channel functional calibration in a parameter-efficient manner.

1) RWKV: We adopt the RWKV's time-mixing and channel-mixing operations as the core of our module. Let $x_t \in \mathbb{R}^C$ denote the token at time t, and let $S(\cdot)$ be a temporal shift operator. We form the shifted input $\tilde{x}_t = S(x_t)$ and compute linear projections:

$$r_t = W_r \tilde{x}_t, \qquad k_t = W_k \tilde{x}_t, \qquad v_t = W_v \tilde{x}_t \tag{1}$$

with $W_r, W_k, W_v \in \mathbb{R}^{C\times C}$. Then, we aggregate values with per-channel exponential time-decay $w \in \mathbb{R}^C$ and a separate "current-token" term via $u \in \mathbb{R}^C$:

$$\tilde{v}_t^{\rightarrow} = \frac{\sum_{i=1}^{t-1} \exp\big(-(t-1-i)w + k_i\big) \odot v_i \;+\; \exp\big(u + k_t\big) \odot v_t}{\sum_{i=1}^{t-1} \exp\big(-(t-1-i)w + k_i\big) \;+\; \exp\big(u + k_t\big)} \tag{2}$$

To leverage full trajectory context as in our implementation, we apply Eq. 2 on the time-reversed sequence as well and combine the two scans:

$$\tilde{v}_t \;=\; \tilde{v}_t^{\rightarrow} \;+\; \tilde{v}_t^{\leftarrow} \tag{3}$$

The time–mixing output is then gated by the receptance and linearly projected:

$$\mathrm{TM}(x_t) \;=\; W_o\big(\sigma(r_t) \odot \tilde{v}_t\big), \qquad W_o \in \mathbb{R}^{C\times C} \tag{4}$$

Complementarily, the channel–mixing branch applies a token-wise gated MLP with squared-ReLU nonlinearity:

$$r'_t = W'_r x_t, \quad k'_t = W'_k x_t, \tag{5}$$

$$\mathrm{CM}(x_t) = \sigma(r'_t) \odot \big(W'_v\,[\mathrm{ReLU}(k'_t)]^2\big) \tag{6}$$

where $W'_r, W'_k, W'_v \in \mathbb{R}^{C\times C}$. Finally, we employ a pre-norm residual, and the output z_t can be described as:

$$y_t = x_t \;+\; \mathrm{CM}\big(\mathrm{LN}_2(x_t)\big), \quad z_t = y_t \;+\; \mathrm{TM}\big(\mathrm{LN}_1(y_t)\big) \tag{7}$$

2) GroupKAN: We briefly recall the Kolmogorov–Arnold Network (KAN) formulation used in our design. A conventional K-layer MLP alternates linear maps and a fixed nonlinearity σ:

$$\mathrm{MLP}(\mathbf{Z}) \;=\; \big(W_{K-1} \circ \sigma \circ W_{K-2} \circ \sigma \circ \cdots \circ W_1 \circ \sigma \circ W_0\big)\mathbf{Z} \tag{8}$$

whereas a K-layer KAN composes K *KAN layers* Φ_k,

$$\mathrm{KAN}(\mathbf{Z}) \;=\; \big(\Phi_{K-1} \circ \Phi_{K-2} \circ \cdots \circ \Phi_1 \circ \Phi_0\big)\mathbf{Z} \tag{9}$$

each Φ_k acting from $\mathbb{R}^{n_k} \to \mathbb{R}^{n_{k+1}}$ via a *matrix of learnable univariate functions* $\{\phi^{(k)}_{q,p}\}_{q,p}$ placed on edges rather than

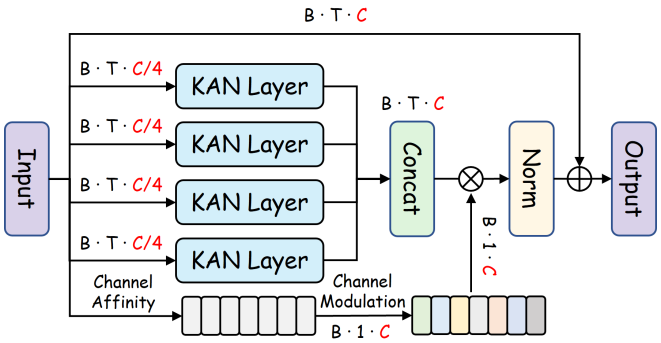

Fig. 3: The architecture of the GroupKAN.

fixed node activations. Writing $\mathbf{Z}_{k+1} = \Phi_k \mathbf{Z}_k$, one has the matrix-of-functions form:

$$\Phi_k = \begin{pmatrix} \phi_{1,1}^{(k)}(\cdot) & \phi_{1,2}^{(k)}(\cdot) & \cdots & \phi_{1,n_k}^{(k)}(\cdot) \\ \phi_{2,1}^{(k)}(\cdot) & \phi_{2,2}^{(k)}(\cdot) & \cdots & \phi_{2,n_k}^{(k)}(\cdot) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{n_{k+1},1}^{(k)}(\cdot) & \phi_{n_{k+1},2}^{(k)}(\cdot) & \cdots & \phi_{n_{k+1},n_k}^{(k)}(\cdot) \end{pmatrix} \tag{10}$$

which eliminates explicit weight matrices and instead learns spline-parameterized univariate functions on edges, a choice that improves parameter efficiency and interpretability while preserving approximation power.

Let $\mathbf{X} \in \mathbb{R}^{B\times T\times C}$ denote a sequence of T time steps with C channels. We partition the channel into G equal groups, $\mathbf{X} = \mathrm{Concat}(\mathbf{X}_1, \ldots, \mathbf{X}_G)$ with $\mathbf{X}_g \in \mathbb{R}^{B\times T\times C/G}$, and process each group by an *independent* KAN operator that shares parameters across time. Concretely, flattening the first two axes, we define the operation process as:

$$\mathbf{Y}_g = \mathrm{KAN}_g(\mathbf{X}_g) \in \mathbb{R}^{B\times T\times C/G}, \quad g = 1, \ldots, G \tag{11}$$

and concatenate the outputs along channels:

$$\mathbf{Y} = \mathrm{Concat}(\mathbf{Y}_1, \ldots, \mathbf{Y}_G) \in \mathbb{R}^{B\times T\times C} \tag{12}$$

To adaptively reweight channels with sequence-aware statistics, we introduce a lightweight *channel affinity modulation* (CAM). Let $\bar{\mathbf{X}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{X}_{[:,t,:]} \in \mathbb{R}^{B\times C}$ denote the temporal pooling outputs; CAM produces a gating vector $a \in R^{B\times C}$ and we broadcast it to $\mathbf{A} \in R^{B\times T\times C}$:

$$a = \sigma\Big(W_2\, \varphi\big(W_1\, \bar{\mathbf{X}}\big)\Big) \in \mathbb{R}^{B\times C} \tag{13}$$

where W_1, W_2 are learned linear maps, φ is a smooth activation (e.g., SiLU), and σ is the logistic sigmoid. The final GroupKAN's output can be denoted as:

$$\widehat{\mathbf{X}} = \mathbf{X} + \mathrm{DropPath}(\mathrm{LN}(\mathbf{A} \odot \mathbf{Y})) \tag{14}$$

where $\odot$ is elementwise multiplication. In practice, we use G=4 and each KAN_g acts on $C/4$ channels. GroupKAN preserves KAN's efficient function approximation and integrates a CAM to highlight task-relevant channels, achieving parameter efficiency without compromising performance.

C. Consistency Flow Matching Objective

We learn a velocity-consistent straight-line flow in the *action* space, conditioned on robot state s and visual representation v. Let $a_{\mathrm{src}} \sim \Gamma_{\mathrm{src}} := \mathcal{N}(0, I)$ and $a_{\mathrm{tar}} \sim \Gamma_{\mathrm{tar}}(\cdot \mid s, v)$, where a_{src} and a_{tar} are action trajectories sampled from the source and target distributions, respectively; Γ_{src} is the standard normal over the action space and Γ_{tar} is the task distribution conditioned on (s, v). For any $t \in [0,1]$, define the linear interpolation between these samples:

$$a_t = (1-t)\, a_{\mathrm{src}} + t\, a_{\mathrm{tar}} \tag{15}$$

A time-conditioned vector field $v_\theta(a_t, t, s, v)$ induces a single-step decoder that advances from t to 1 via explicit Euler with remaining step size $(1-t)$:

$$f_\theta(t, a_t, s, v) = a_t + (1-t)\, v_\theta(a_t, t, s, v) \tag{16}$$

Here, θ denotes all learnable parameters of the velocity (flow) network v_θ, and f_θ depends on θ only through v_θ.

a) Consistency objective: Consistency Flow Matching (CFM) enforces that decoding from two nearby times t and $t+\Delta t$ yields the same endpoint, and that their instantaneous velocities agree. With an exponential moving average (EMA) and a small $\Delta t > 0$, we minimize:

$$\mathscr{L}_{\mathrm{end}} = \mathbb{E}_{t\sim\mu}\, \mathbb{E}_{a_t, a_{t+\Delta t}} \Big[\big\| f_\theta(t, a_t, s, v) - f_{\theta^-}(t+\Delta t, a_{t+\Delta t}, s, v) \big\|_2^2 \Big] \tag{17}$$

$$\mathscr{L}_{\mathrm{vel}} = \mathbb{E}_{t\sim\mu}\, \mathbb{E}_{a_t, a_{t+\Delta t}} \Big[\big\| v_\theta(a_t, t, s, v) - v_{\theta^-}(a_{t+\Delta t}, t+\Delta t, s, v) \big\|_2^2 \Big] \tag{18}$$

$$\mathscr{L}_{\mathrm{CFM}} = \mathscr{L}_{\mathrm{end}} + \alpha\, \mathscr{L}_{\mathrm{vel}} \tag{19}$$

where μ is uniform on $[0, 1-\Delta t]$, α denotes positive scalar, Δt represents the time interval, and θ^- is the running average of past θ values.

b) Multi-segment extension: To relax global consistency and improve expressivity, we partition $[0,1]$ into K uniform segments. For segment $i \in \{0, \ldots, K-1\}$, we optimize the segment-wise consistency:

$$f_\theta^{(i)}(t, a_t, s, v) = a_t + \big((i+1)/K - t\big)\, v_\theta^{(i)}(a_t, t, s, v) \tag{20}$$

$$a_i(t) = \big\| f_\theta^{(i)}(t, a_t, s, v) - f_{\theta^-}^{(i)}(t+\Delta t, a_{t+\Delta t}, s, v) \big\|_2^2 \tag{21}$$

$$b_i(t) = \big\| v_\theta^{(i)}(a_t, t, s, v) - v_{\theta^-}^{(i)}(a_{t+\Delta t}, t+\Delta t, s, v) \big\|_2^2 \tag{22}$$

$$\mathscr{L}_{\mathrm{MFM}} = \sum_{i=0}^{K-1} \lambda_i\, \mathbb{E}_{t\sim\mu_i}\, \mathbb{E}_{a_t, a_{t+\Delta t}} \big[a_i(t) + \alpha\, b_i(t) \big] \tag{23}$$

where μ_i is uniform on $\big[i/K, (i+1)/K - \Delta t\big]$ and $\{\lambda_i\}$ balance segment importance. In practice we use K=2 for a strong efficiency–quality trade-off.

D. Action Consistency Regularization (ACR)

To explicitly anchor the *one-step* decode at the terminal time, we introduce an action consistency regularizer that ties the one-step decode $f_\theta(t,a_t,s,v)$ at $t\to 1$ to expert actions over a task-relevant control window.

Given the interpolation sample a_t and conditions $c{:=}(s,v)$, we form a single-step decode using Eq. 16:

$$\hat{a}_1 \;=\; f_\theta(t,a_t,s,v) \;=\; a_t + (1-t)\,v_\theta(a_t,t,s,v) \tag{24}$$

Let the control window $W=\{u_0,\ldots,u_0{+}H{-}1\}$ denote the action horizon used for behavior matching, with $u_0 = n_{\text{obs}}-1$ and H the horizon length. Let $\mathscr{S}_W$ be the selector that extracts indices W from a full action trajectory. We penalize the mean-squared deviation between the decoded actions and expert actions on W (available *only during training*):

$$\mathscr{L}_{\text{ACR}} \;=\; \frac{1}{|W|}\sum_{u\in W}\Big\|\big(\mathscr{S}_W\hat{a}_1\big)_u - a^\star_u\Big\|_2^2 \tag{25}$$

Intuitively, Eq. 24 performs a single-step decode from (a_t,t) to $t{=}1$ via Eq.16; Eq. 25 enforces behavioral consistency with expert actions over W, biasing the learned vector field toward action-faithful solutions without extra network evaluations.

E. Overall Objective and Inference

Our final training loss combines the flow matching objective and the ACR term:

$$\mathscr{L} \;=\; \mathscr{L}_{\text{MFM}} \;+\; \lambda_{ACR}\,\mathscr{L}_{ACR} \tag{26}$$

where $\lambda_{\text{ACR}}\geq 0$ (set to 1 in our experiments). At test time, we draw $a_0\sim\mathcal{N}(0,I)$ and perform the one-step decode $a_1 = \Pi_a\big(x_t+(1-t)\,v_\theta(x_t,t,c)\big)$ with $t\in(0,1)$, yielding real-time action sequences while preserving high task success rates due to the ACR constraint.

IV. Experiment

A. Datasets

We evaluate KAN-We-Flow on three benchmarks to verify its effectiveness, including 3 tasks in Adroit [36], 4 tasks in Dexart [37], and 34 tasks in Meta-World [38]. Following previous work [7], [11], [12], we adopt the same data collection method.

B. Baselines

To demonstrate the effectiveness of KAN-We-Flow, we compare it against both diffusion-based and flow-matching methods. For diffusion-based baselines, we include 3D diffusion policies (DP3 [7], Simple-DP3 [7], and MambaPolicy [8]) and 2D diffusion policies (DP [9], BCRNN [39], and IBC [40]). For flow-matching baselines, we consider Adaflow [41], FlowPolicy [11], and MP1 [12]. Because the expert demonstrations are generated stochastically, we re-implemented DP3 [7], MambaPolicy [8], and FlowPolicy [11] under the same environment and expert data as our method; we denote these runs with † to indicate our reimplementations and ensure a fair comparison.

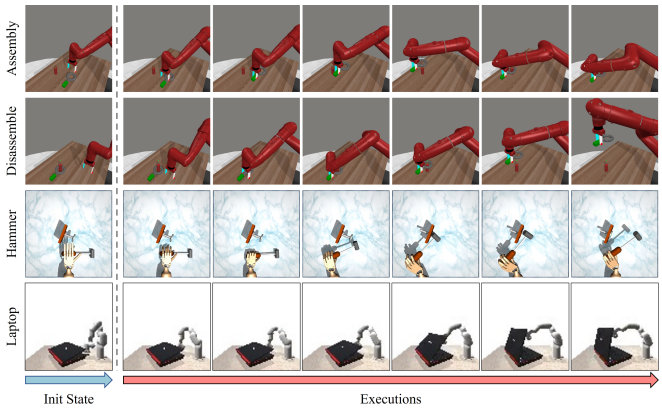

Fig. 4: Visualization of manipulation results. We evaluate on three datasets: Meta-World, Adroit, and DexArt. We illustrate representative rollouts from Meta-World (Assembly, Disassemble), Adroit (Hammer), and DexArt (Laptop). During interaction, the KAN-We-Flow policy predicts future action sequences and continues issuing actions until the task is successfully completed.

C. Implementation Details

We generate ten expert demonstrations for training on Adroit [36], Dexart [37], and Meta-World [38]. Like the previous works [7], [8], the image size is cropped to a resolution of 84×84 pixels, and the point cloud is downsampled to 512 points for Adroit or 1024 points for Meta-World using farthest point sampling (FPS). We set the prediction horizon to 4, the observation length to 2, and the action prediction length to 3. We adopt an AdamW optimizer to update the weights during the training phase, with a learning rate of $1e-4$ and a batch size of 128. The model is trained for 3,000 epochs. We use an EMA decay of 0.95. During training, both actions and states are normalized to the range [-1, 1]; actions are de-normalized before execution. All implementations, including ours and the baselines, are built in PyTorch and evaluated on a single NVIDIA RTX 3090 GPU.

D. Evaluation Metrics

We report top-1, top-3, and top-5 success rates (*SR*1, *SR*3, *SR*5), computed as the average of the highest 1, 3, and 5 trial outcomes per task, respectively. For each domain, we use three random seeds $(0,42,100)$ and report the mean and standard deviation across the seeds.

E. Comparisons with the State-of-the-Art

We compare KAN-We-Flow against both diffusion-based (DP3, Simple-DP3, MambaPolicy, Diffusion Policy, BCRNN, IBC) and flow-based (Adaflow, FlowPolicy, MP1) policies on Adroit, DexArt, and Meta-World. As summarized in Table I, our method achieves the best overall success rates across all domains and difficulty tiers. On *Adroit*, KAN-We-Flow raises Door and Pen to 83.0±2.0 and 68.0±2.5, outperforming FlowPolicy (62.0±5.0, 55.0±13.0) and DP3 (65.0±3.3, 42.3±1.5). On *DexArt*, it improves over FlowPolicy across Faucet, Toilet, Bucket, and Laptop. On

TABLE I: Quantitative comparison of baselines in simulated environments. We compare KAN-We-Flow against diffusion-based and flow-matching methods on the Adroit, Meta-World, and DexArt datasets. † indicates results reproduced by us for fair comparison. KAN-We-Flow achieves the best performance across all domains.

Method	Adroit			DexArt				Meta-World			
	Hammer	Door	Pen	Laptop	Faucet	Toilet	Bucket	Easy (21)	Medium (4)	Hard (4)	Very Hard (5)
Diffusion Policy [9]	48.0±17.0	50.0±5.0	25.0±4.0	69.0±4.0	23.0±8.0	58.0±2.0	46.0±1.0	50.7±6.1	11.0±2.5	5.25±2.5	22.0±5.0
BCRNN [39]	8.0±14.0	0.0±0.0	8.0±1.0	29.0±12.0	26.0±2.0	38.0±10.0	24.0±11.0	-	-	-	-
IBC [40]	0.0±0.0	0.0±0.0	10.0±1.0	1.0±1.0	7.0±2.0	15.0±1.0	0.0±0.0	-	-	-	-
DP3 [7] †	100.0±0.0	65.0±3.3	42.3±1.5	78.0±3.5	32.0±3.5	75.3±2.5	29.0±2.3	88.3±3.2	45.5±8.5	33.7±8.7	42.0±9.0
Simple DP3 [7] †	98.0±2.0	50.0±2.3	40.0±1.3	75.0±2.3	30±2.0	70.0±2.0	25.0±1.3	87.0±2.0	43.0±7.0	30.0±6.0	36.0±6.0
MambaPolicy [8] †	100.0±0.0	70.0±1.0	42.0±2.3	80.0±1.0	35.0±2.0	77.0±1.0	28.0±1.0	89.0±1.0	46.0±7.5	34.3±9.3	43.0±8.0
Adaflow [13]	45.0±11.0	27.0±6.0	18.0±6.0	-	-	-	-	50.4±7.0	14.0±6.0	8.0±5.0	25.0±6.0
FlowPolicy [11] †	100.0±0.0	62.0±5.0	55.0±13.0	81.0±1.0	34.0±4.0	77.0±3.0	30.0±3.3	85.0±3.3	59.0±8.0	42.0±5.0	53.0±6.0
MP1 [12]	100.0±0.0	69.0±2.0	58.0±5.0	-	-	-	-	88.2±1.1	68.0±3.1	58.1±5.0	67.2±2.7
Ours	**100.0±0.0**	**83.0±2.0**	**68.0±2.5**	**85.0±2.5**	**42.5±2.5**	**82.0±2.0**	**35.0±2.3**	**92.0±1.0**	**72.0±1.3**	**63.3±1.3**	**71.3±1.0**

TABLE II: Comparison of inference time across methods on the Meta-World benchmark. Because of the multi-step denoising procedure, diffusion-based methods are slower than flow-based methods. KAN-We-Flow achieves state-of-the-art inference speed across all subtasks.

Method	Adroit/ms			Meta-World/ms			
	Hammer	Door	Pen	Easy (21)	Medium (4)	Hard (4)	Very Hard (5)
DP3 [7]	129.5±13.9	141.3±14.0	130.0±10.2	130.0±10.5	132.0±10.3	130.0±10.5	135.0±10.5
Simple DP3 [7]	103.1±11.4	111.3±10.2	115.0±3.0	91.9±8.6	98.3±9.1	101.3±9.7	103.8±10.2
FlowPolicy [11]	15.3±1.1	13.2±4.0	20.0±0.6	12.0±1.4	12.2±1.5	13.5±1.4	14.5±1.6
Ours	**9.5±2.3**	**8.0±2.5**	**10.1±0.3**	**10.1±1.0**	**6.7±0.1**	**10.3±1.1**	**11.8±1.1**

Meta-World, it exceeds both MP1 and FlowPolicy across *Easy/Medium/Hard/Very Hard*; for example, 92.0±1.0 vs. 88.2±1.1 (MP1) and 85.0±3.3 (FlowPolicy) on Easy, and 71.3±1.0 vs. 67.2±2.7 (MP1) and 53.0±6.0 (FlowPolicy) on Very Hard. These aggregate gains are visualized in Fig. 5 through $SR1/SR3/SR5$ bars, highlighting consistent improvements not only at top-1 but also at the top-K operating points used in prior work.

F. Discussion

We attribute the accuracy gains to two complementary design choices: (i) the *RWKV–KAN* UNet, which captures long-horizon temporal dependencies and per-channel functional calibration with far fewer parameters than previous UNet-style stacks; and (ii) the *Action Consistency Regularization (ACR)*, which anchors the one-step decode to expert behaviors and reduces drift, particularly beneficial on longer-horizon, harder tasks. Overall, these results validate the central thesis from the introduction: replacing heavy UNet-style backbones with RWKV–KAN UNet and adding a lightweight ACR term yields a policy that is both more *accurate* and *deployable* across diverse manipulation benchmarks.

G. Efficiency Analysis

Beyond accuracy, KAN-We-Flow is markedly more efficient in parameters and runtime. Table III shows that our model uses only **33.6M** parameters and **0.03G** FLOPs with **101.2 MB** GPU memory, cutting parameters by **86.8%** vs. DP3 (255.1M) and still improving success on challenging Meta-World tasks such as Stick-Pull (VH), Disassemble (VH), and Pick-Out-Of-Hole (H). Inference speed mirrors these savings: as reported in Table III, KAN-We-Flow runs in

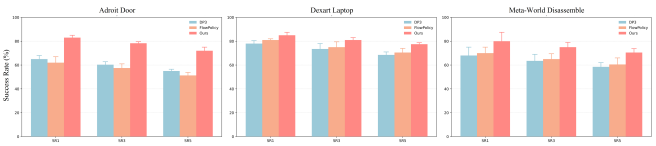

Fig. 5: Visualization of success rates. We plot top-1, top-3, and top-5 success rates (SR1, SR3, SR5), computed as the average of the highest 1, 3, and 5 trial outcomes per task; KAN-We-Flow achieves superior performance across tasks.

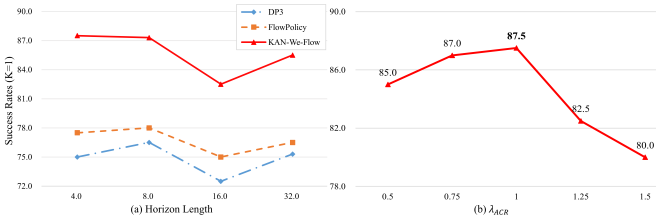

Fig. 6: Ablations on horizon and hyperparameters. **(a)** Varying the prediction horizon: *KAN–We–Flow* keeps stable and consistently outperforms DP3 and FlowPolicy. **(b)** λ_{ACR} weight: performance peaks at $\lambda_{\text{ACR}}=1$; smaller values under-constrain end-point alignment, larger ones mildly over-regularize.

8.0–10.1 ms on Adroit Door/Pen/Hammer and **6.7–11.8 ms** across Meta-World tiers, comfortably supporting ∼100 Hz control. This is substantially faster than diffusion baselines (e.g., DP3 ∼103–141 ms) and improves over FlowPolicy's already fast flow-matching inference (e.g., Door 13.2±4.0 ms → **8.0**±2.5 ms). The combination of a one-step CFM decoder and the RWKV–KAN UNet therefore delivers both latency and memory wins without sacrificing accuracy. In short, KAN-We-Flow meets practical needs by enabling real-time control with compact models that fit edge compute budgets. It also outperforms prior SOTA methods, achieving precisely the efficiency–accuracy balance we targeted.

H. Ablation Study

1) Ablations on Key Components: We first assess the contribution of each proposed component on Adroit. Starting from the Baseline, progressively inserting RWKV time-mixing, RWKV channel-mixing, GroupKAN, and the Ac-

TABLE III: We compare parameter efficiency across DP3, FlowPolicy, MambaPolicy, and KAN-We-Flow. In the challenging Meta-World settings, our method achieves stronger results with 86.8% fewer parameters compared with DP3. We abbreviate the Very Hard and Hard difficulty levels as VH and H.

Method	Param (M)	FLOPs (G)	GPU Usage (MB)	Hard Environment in Meta-World		
				Stick-Pull (VH)	Disassemble (VH)	Pick-Out-Of-Hole (H)
DP3 [7]	255.1	0.3	996.1	82.6	75.0	45.0
FlowPolicy [11]	255.5	0.4	980.2	83.4	78.5	47.5
MambaPolicy [8]	47.9	0.03	137.7	86.7	81.7	48.3
Ours	**33.6**	**0.03**	**101.2**	**90.2**	**87.5**	**52.5**

TABLE IV: Ablation study of key components of KAN-We-Flow in the Adroit Door environment. The results highlight the necessity of each proposed component.

Baseline	RWKV Time-Mixing	RWKV Channel-Mixing	GroupKAN	ACR	Adroit-Pen		
					*SR*1	*SR*3	*SR*5
✓					59.5±1.5	51.5±1.3	45.5±2.5
✓	✓				62.0±1.3	52.5±2.3	48.3±1.3
✓	✓	✓			64.3±2.0	55.5±2.0	50.5±0.3
✓	✓		✓		65.0±3.3	58.3±1.3	53.3±2.3
✓		✓	✓		65.5±2.3	58.0±1.5	54.0±2.0
✓	✓	✓	✓	✓	**68.0±2.5**	**61.3±1.0**	**56.5±1.5**

TABLE V: Ablation of architectural components within the RWKV–KAN block on Adroit–Door. We evaluate replacements of RWKV with CNN/MHSA/Mamba-V1/V2 and GroupKAN with KAN/MLP. "r/w" denotes "replaced with".

Method	Adroit-Door		
	*SR*1	*SR*3	*SR*5
w/o RWKV, r/w CNN block	66.5±2.5	59.3±3.0	55.0±3.0
w/o RWKV, r/w MHSA block [42]	68.0±5.0	61.3±2.0	58.3±2.3
w/o RWKV, r/w Mamba-V1 [43]	70.5±1.5	65.5±3.5	62.3±2.0
w/o RWKV, r/w Mamba-V2 [43]	71.3±1.0	67.5±2.0	64.3±1.0
w/o GroupKAN, r/w KAN [23]	81.0±0.5	76.0±0.5	70.5±0.5
w/o GroupKAN, r/w MLP	80.3±1.3	75.0±1.0	69.5±1.5
Ours	**83.0±2.0**	**78.3±1.3**	**72.0±3.0**

tion Consistency Regularization (ACR) yields steady improvements in $SR1/SR3/SR5$ (Table IV). Concretely, adding RWKV's time-mixing already improves robustness, and further introducing the channel-mixing branch strengthens long-range temporal propagation; replacing the heavy per-channel MLP with GroupKAN consistently lifts all top-K operating points by enabling compact, spline-based functional calibration; finally, the auxiliary ACR term provides the last increment by explicitly anchoring the one-step decode to expert behavior without extra inference steps. The full model (RWKV + GroupKAN + ACR) attains the best performance, substantiating the complementary roles of (i) linear-time sequence mixing (RWKV), (ii) parameter-efficient per-channel calibration (GroupKAN), and (iii) horizon-end anchoring (ACR).

2) Ablation of Architectural Components within the RWKV–KAN Block: We next replace the two core submodules in our RWKV-KAN block with strong alternatives to isolate where the gains come from (Table V, Adroit–Door). Swapping RWKV for a CNN or an MHSA (Transformer) block improves over the baseline but remains notably behind our design, indicating that linear-time receptance-weighted mixing is particularly well-suited to action-trajectory modelling under tight latency budgets. Replacing RWKV with state-space variants (Mamba-V1/V2) yields larger gains than CNN/MHSA and narrows the gap, yet our RWKV block retains the top scores in $SR1/SR3/SR5$, suggesting that the receptance-gated, bidirectional scan we adopt better preserves long-horizon consistency in this setting. On the calibration side, substituting GroupKAN with a plain KAN or with an MLP confirms that learnable spline functions help, but the groupwise formulation with channel-affinity modulation is consistently stronger.

I. Ablations on Horizon Lengths and Hyperparameters

Finally, we vary the action-prediction horizon to probe long-term dependency modelling (Fig. 6 a). Across short-to-long horizons, KAN–We–Flow maintains stable $SR1/SR3/SR5$ and outperforms DP3 [7] and FlowPolicy [11] at every setting, indicating that (i) RWKV's linear-time temporal mixing effectively propagates context over extended windows and (ii) ACR's end-point anchoring mitigates exposure bias as the decode spans longer segments. The flat performance–horizon curve further suggests that our design achieves the desired efficiency–accuracy balance: it scales to longer horizons without resorting to multi-step refinement, while retaining real-time, one-step inference.

We ablate the hyperparameter λ_{ACR} over a broad range while keeping other settings fixed. As shown in Fig. 6 b, performance peaks at $\lambda_{\text{ACR}}=1$; smaller values under-penalise end-point drift and larger values slightly over-regularise, with a relatively flat neighbourhood around the optimum indicating robustness.

V. Conclusion

In this work, we introduced KAN-We-Flow, a lightweight flow-matching visuomotor policy built on a synergistic RWKV–KAN UNet with a simple Action Consistency Regularization (ACR) term. Across Adroit, DexArt, and Meta-World, our model achieved state-of-the-art success rates while delivering practical efficiency: an 86.8% parameter reduction relative to UNet-style diffusion baselines and single-step inference in the 8–11 ms range, enabling 100 Hz control. Ablations confirm that (i) RWKV's linear-time temporal mixing and (ii) GroupKAN's parameter-efficient, spline-based per-channel calibration are complementary, and that (iii) ACR stabilises one-step decoding, particularly for longer horizons. We believe these results demonstrate a clear path toward accurate, real-time, edge-deployable robot policies without heavy UNets. Future work will extend our

approach to real-robot evaluations, richer multi-view sensing and language conditioning, and contact-rich, long-horizon tasks where safety and sim-to-real robustness are paramount.

REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[2] B. Wang, G. Wu, T. Pang, Y. Zhang, and Y. Yin, "Diffail: Diffusion adversarial imitation learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 14, 2024, pp. 15 447–15 455.

[3] C. Wang, H. Fang, H.-S. Fang, and C. Lu, "Rise: 3d perception makes real-world robot imitation simple and effective," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 2870–2877.

[4] A. Agarwal, S. Uppal, K. Shaw, and D. Pathak, "Dexterous functional grasping," *arXiv preprint arXiv:2312.02975*, 2023.

[5] S. Haldar, J. Pari, A. Rai, and L. Pinto, "Teach a robot to fish: Versatile imitation from one minute of demonstrations," *arXiv preprint arXiv:2303.01497*, 2023.

[6] G. Lu, Z. Gao, T. Chen, W. Dai, Z. Wang, W. Ding, and Y. Tang, "Manicm: Real-time 3d diffusion policy via consistency model for robotic manipulation," *arXiv preprint arXiv:2406.01586*, 2024.

[7] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," *arXiv preprint arXiv:2403.03954*, 2024.

[8] 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," *arXiv preprint arXiv:2409.07163*, 2024.

[9] 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*, p. 02783649241273668, 2023.

[10] R. Yang, G. Chen, C. Wen, and Y. Gao, "Fp3: A 3d foundation policy for robotic manipulation," *arXiv preprint arXiv:2503.08950*, 2025.

[11] Q. Zhang, Z. Liu, H. Fan, G. Liu, B. Zeng, and S. Liu, "Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 14, 2025, pp. 14 754–14 762.

[12] J. Sheng, Z. Wang, P. Li, and M. Liu, "Mp1: Meanflow tames policy learning in 1-step for robotic manipulation," *arXiv preprint arXiv:2507.10543*, 2025.

[13] X. Hu, Q. Liu, X. Liu, and B. Liu, "Adaflow: Imitation learning with variance-adaptive flow-based policies," *Advances in Neural Information Processing Systems*, vol. 37, pp. 138 836–138 858, 2024.

[14] J. Postels, M. Danelljan, L. Van Gool, and F. Tombari, "Maniflow: Implicitly representing manifolds with normalizing flows," in *2022 International Conference on 3D Vision (3DV)*. IEEE, 2022, pp. 84–93.

[15] H. Zhi, P. Chen, S. Zhou, Y. Dong, Q. Wu, L. Han, and M. Tan, "3dflowaction: Learning cross-embodiment manipulation from 3d flow world model," *arXiv preprint arXiv:2506.06199*, 2025.

[16] L. Yang, L. Li, X. Xin, Y. Sun, Q. Song, and W. Wang, "Large-scale person detection and localization using overhead fisheye cameras," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 19 961–19 971.

[17] P. Cao, F. Zhou, L. Yang, T. Huang, and Q. Song, "Image is all you need to empower large-scale diffusion models for in-domain generation," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 18 358–18 368.

[18] P. Cao, F. Zhou, Q. Song, and L. Yang, "Controllable generation with text-to-image diffusion models: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[19] T. Chen, X. Zhou, Z. Tan, Y. Wu, Z. Wang, Z. Ye, T. Gong, Q. Chu, N. Yu, and L. Lu, "Zig-rir: Zigzag rwkv-in-rwkv for efficient medical image segmentation," *IEEE Transactions on Medical Imaging*, 2025.

[20] Y. Duan, W. Wang, Z. Chen, X. Zhu, L. Lu, T. Lu, Y. Qiao, H. Li, J. Dai, and W. Wang, "Vision-rwkv: Efficient and scalable visual perception with rwkv-like architectures," *arXiv preprint arXiv:2403.02308*, 2024.

[21] Z. Fei, M. Fan, C. Yu, D. Li, and J. Huang, "Diffusion-rwkv: Scaling rwkv-like architectures for diffusion models," *arXiv preprint arXiv:2404.04478*, 2024.

[22] X. Zhou and T. Chen, "Bsbp-rwkv: Background suppression with boundary preservation for efficient medical image segmentation," in *Proceedings of the 32nd ACM International Conference on Multimedia*, 2024, pp. 4938–4946.

[23] C. Li, X. Liu, W. Li, C. Wang, H. Liu, Y. Liu, Z. Chen, and Y. Yuan, "U-kan makes strong backbone for medical image segmentation and generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 5, 2025, pp. 4652–4660.

[24] S. Somvanshi, S. A. Javed, M. M. Islam, D. Pandit, and S. Das, "A survey on kolmogorov-arnold network," *ACM Computing Surveys*, 2024.

[25] A. A. Aghaei, "rkan: Rational kolmogorov-arnold networks," *arXiv preprint arXiv:2406.14495*, 2024.

[26] Z. Hou, T. Zhang, Y. Xiong, H. Pu, C. Zhao, R. Tong, Y. Qiao, J. Dai, and Y. Chen, "Diffusion transformer policy," *arXiv preprint arXiv:2410.15959*, 2024.

[27] 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*, 2024, pp. 18 081–18 090.

[28] L. Yang, L. Li, J. Wei, P. Cao, Y. Wang, and W. Wang, "Large-scale omnidirectional person positioning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[29] T. Feng, X. Wang, Y.-G. Jiang, and W. Zhu, "Embodied ai: From llms to world models," *IEEE Circuits and Systems Magazine*, 2025.

[30] T. Feng, X. Wang, Z. Zhou, R. Wang, Y. Zhan, G. Li, Q. Li, and W. Zhu, "Evoagent: Agent autonomous evolution with continual world model for long-horizon tasks," *arXiv preprint arXiv:2502.05907*, 2025.

[31] Y. Su, N. Liu, D. Chen, Z. Zhao, K. Wu, M. Li, Z. Xu, Z. Che, and J. Tang, "Freqpolicy: Efficient flow-based visuomotor policy via frequency consistency," *arXiv preprint arXiv:2506.08822*, 2025.

[32] N. Gkanatsios, J. Xu, M. Bronars, A. Mousavian, T.-W. Ke, and K. Fragkiadaki, "3d flowmatch actor: Unified 3d policy for single- and dual-arm manipulation," *arXiv preprint arXiv:2508.11002*, 2025.

[33] S. Dasari, O. Mees, S. Zhao, M. K. Srirama, and S. Levine, "The ingredients for robotic diffusion transformers," *arXiv preprint arXiv:2410.10088*, 2024.

[34] Q. Wang, Y. Sun, E. Lu, Q. Zhang, and Y. Zeng, "Mtdp: A modulated transformer based diffusion policy model," *arXiv preprint arXiv:2502.09029*, 2025.

[35] Y. Ge, Z. Chen, M. Yu, Q. Yue, R. You, and L. Zhu, "Mambatsr: You only need 90k parameters for traffic sign recognition," *Neurocomputing*, vol. 599, p. 128104, 2024.

[36] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[37] C. Bao, H. Xu, Y. Qin, and X. Wang, "Dexart: Benchmarking generalizable dexterous manipulation with articulated objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 190–21 200.

[38] 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*. PMLR, 2020, pp. 1094–1100.

[39] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.

[40] 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*. PMLR, 2022, pp. 158–168.

[41] X. Hu, Q. Liu, X. Liu, and B. Liu, "Adaflow: Imitation learning with variance-adaptive flow-based policies," *Advances in Neural Information Processing Systems*, vol. 37, pp. 138 836–138 858, 2024.

[42] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[43] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.