IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

KAN Policy: Learning Efficient and Smooth Robotic Trajectories via Kolmogorov-Arnold Networks

Zikang Chen[1], Fei Gao[2], Ziya Yu[3], and Peng Li[4]

***Abstract*—Modern robotic visuomotor policy learning has witnessed significant progress through Diffusion Policy (DP) frameworks built upon *Convolutional Neural Networks* (CNNs) and Transformers. Despite their empirical success, these architectures remain fundamentally constrained by their relatively discrete computational nature, inherently limiting their capacity to generate efficient and smooth motion trajectories. To address this challenge, we introduce *Kolmogorov-Arnold Networks* (KANs) into Diffusion Policy learning. The proposed *KAN Policy* (KP) leverages KANs' intrinsic continuity through learnable base-parameterized activation functions, thereby producing continuous trajectories with shorter execution time and fewer jerks. Specifically, we design a novel *Embedding KAN* (Emb-KAN) for CNN-based models, which preserves structural continuity in high-dimensional latent spaces through adaptive spline embeddings. Besides, we apply Group-KAN to Transformer-based models for learning continuous representations. Across main simulation experiments, KP achieves average improvements of 6.06%, 8.03%, and 26.4% in terms of success rate, execution time, and smoothness, respectively. Similarly, in real-world experiments, KP achieves average improvements of 53.8%, 7.89%, and 29.4% across the same metrics.**

***Index Terms*—Deep Learning Methods,Motion and Path Planning,Learning from Demonstration.**

I. Introduction

POLICY learning from demonstrations, typically formulated as a supervised regression problem mapping observations to actions, has shown remarkable success in diverse robotic applications [1] [2] [3]. Among generative approaches, Diffusion Policy (DP) frameworks [4] [5] [6] [7] have emerged as particularly powerful for robotic visuomotor policy learning, outperforming conventional methods. Current DP implementations primarily employ *Convolutional Neural Networks* (CNNs) [8] or Transformers [9], architectures that inherently operate on discrete data representations. This discretization mechanism fractures feature continuity, thereby imposing fundamental limitations. As illustrated by the yellow trajectory in Fig. 1, conventional CNN-based DP models (DP-C) generate abrupt motion transitions rather than smooth trajectory. Such jerks not only extend task execution time but may also pose safety risks in physical deployments [10].

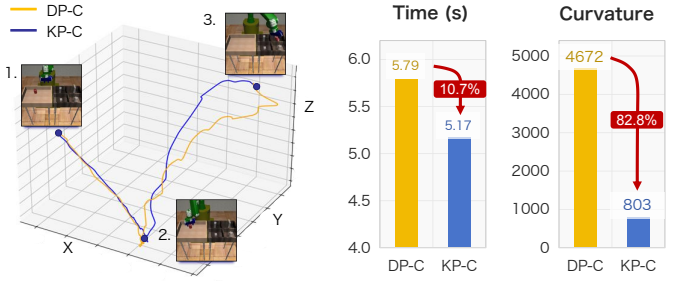

Fig. 1: We compare KP and DP in "Can" [2] as an example. The left curve diagram shows the spatial motion trajectory required to complete the task, and comparison demonstrates that KP-C produces smoother trajectory than DP-C, which has fewer jerks. Additionally, KP-C reduces execution time by 10.7% (middle) and achieves a 82.8% lower curvature than DP-C (right).

Recent advances in Kolmogorov-Arnold Networks (KANs) [11] have demonstrated remarkable potential across a wide range of computer vision applications, including medical image segmentation [12], hyperspectral image analysis [13], and foundational vision models [14]. Unlike conventional CNNs and Transformers, KANs implement a fundamental architectural paradigm shift, they replace fixed linear operations [15] with learnable connections parameterized by adaptive basis functions—notably spline curves [16]. This design proves particularly advantageous for robotic control systems, where spline-based trajectory optimization has long been established as one of the widely adopted methods for ensuring C^2-continuous motions [17] [18] [19]. Besides, KANs also excel by breaking down complex functions into simpler and low-dimensional parts, which makes them efficient and adept at capturing detailed patterns in data [12]. Given these properties, KANs are increasingly recognized as a highly promising candidate for robotic policy learning.

In this paper, we introduce KANs to DP models, termed *KAN Policy* (KP). Specifically, we develop a novel *Embedding KAN* (Emb-KAN) for CNN-based DP models, called KAN Policy-C (KP-C), to feature continuously structural representations, while adapting Group-KAN [14] to Transformer-based counterparts, named KAN Policy-T (KP-T). As demonstrated by the blue trajectory and the two histplots comparing in Fig. 1, KP-C produces continuous trajectory with fewer jerks and shorter execution time. To validate the effectiveness of KP, we conduct seven simulation tasks and three real-world

Manuscript received: March 31, 2025; Revised June 29, 2025; Accepted August 28, 2025.

This paper was recommended for publication by Editor Aniket Bera upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by Science and Technology Major Project of Jiangsu Province (No.BG2024041) and Nanjing Science and Technology Plan (No. Y23002ZX01), and in part by the National Natural Science Foundation of China under Grant 62571395. *(Corresponding author: Peng Li.)*

[1]Zikang Chen and [4]Peng Li are with University of Chinese Academy of Sciences, Nanjing, Nanjing 210008, China; Nanjing Institute of Software Technology, Nanjing 211135, China; Institute of Software Chinese Academy of Sciences, Beijing 100190, China; and the University of Chinese Academy of Sciences, Beijing 100049, China; (e-mails: chenzikang23@mails.ucas.ac.cn; lipeng@iscas.ac.cn).

[2]Fei Gao is with the Hangzhou Research Institute, Xidian University, Hangzhou 311200, China (e-mail: fgao@xidian.edu.cn).

[3]Ziya Yu is with Hohai University, Nanjing 210024, China; Nanjing Institute of Software Technology, Nanjing 211135, China; and the University of Chinese Academy of Sciences, Nanjing, Nanjing 210008, China (e-mail: yuziya24@mails.ucas.ac.cn).

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

experiments, with diverse complexities. In main simulation, KP achieves average improvements of 6.06% in success rate, 8.03% in execution time, and 26.4% in trajectory smoothness. Real-world experiments further demonstrate 53.8% higher success rate, 7.89% faster execution, and 29.4% smoother motions compared to baselines. The ablation and generalization studies further verify the effectiveness and generalization capacity of KP in learning efficient and smooth robotic policies.

In summary, our **primary contributions** are: (1) To the best of our knowledge, *our work is the first to introduce Kolmogorov-Arnold Networks into Diffusion Policy models for robotic policy learning*, achieving simultaneous gains in trajectory effectiveness, smoothness, and overall performance. (2) Technically, we design a novel Emb-KAN module for CNN-based Diffusion Policy models, and we apply Group-KAN to Transformer-based Diffusion Policy models and further generalize. (3) Substantial experiments demonstrate the effectiveness of our KAN Policy in both simulated and real-world robot control tasks, validating its practical applicability and robustness across different environments.

II. Related Work

A. Kolmogorov-Arnold Networks

Kolmogorov-Arnold representation theorem [20] [21] [22] has showed that any continuous function of multiple variables $f(x_1, \ldots, x_n)$ can be expressed as a combination of continuous functions of a single variable and addition $\sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$, where $\phi_{q,p}$ and Φ_q are univariate functions. Inspired by the theorem, *Kolmogorov-Arnold Networks* (KANs) [11] offer a novel neural network architecture that enhances interpretability and accuracy. Unlike traditional neural networks [15], KANs feature activation functions at the network edges and replace weight parameters with learnable univariate functions, a Kolmogorov–Arnold layer with d_{in}-dimensional inputs $\mathbf{x}$ and d_{out}-dimensional outputs can be simply described as $f(\mathbf{x}) = f(x_1, \ldots, x_{d_{\text{in}}}) = \left[\sum_{i=1}^{d_{\text{in}}} \phi_{i,1}(x_i) \cdots \sum_{i=1}^{d_{\text{in}}} \phi_{i,d_{\text{out}}}(x_i) \right]$, $\phi(x)$ often parameterized by spline functions [16]. This design shows potential in robotics but has seen limited application so far. Recently, the integration of KANs with other neural networks has proven to be beneficial. In the realm of *Convolutional Neural Networks* (CNNs) [8], U-KAN [12] has introduced a token-based module by incorporating KANs into the traditional U-Net [23] architecture, leading to significant improvements in segmentation accuracy, particularly in boundary detail processing and small target detection. When it comes to Transformers [24], KAT [14] has effectively extended the advantages of KANs to natural language processing and sequence modeling by utilizing rational functions [25] [26] [27] in place of B-splines in $\phi(x)$ and sharing the parameters of these functions, while also enhancing computational efficiency. Despite these advancements, the integration of KANs has not yet been fully extended to robotic control tasks. There have been attempts to introduce KANs into reinforcement learning for robotic applications [28], but such approaches remain limited and have not been sufficiently generalized for broader robotic control scenarios. Our work aims to address these gaps by building on the aforementioned integration concepts and employing KANs in a modular form to enhance their applicability and effectiveness in robotic control tasks.

B. Diffusion Policy

Traditional policy learning approaches [1] [2] [3] and generative models [29] [30] often exhibit limited expressiveness in capturing complex action distributions, particularly under multimodal uncertainties. To address these limitations, *Diffusion Policy* (DP) [4] introduces a novel paradigm that integrates *Denoising Diffusion Probabilistic Models* (DDPMs) [31] into robotic policy learning. In this framework, action sequence's generation employs a conditional diffusion process comprising: (1) A forward process that incrementally adds noise to ground-truth actions, transforming it into Gaussian noise. (2) A revserse denoising process that reconstructs actions from noise conditioned on current observation, which is realized via *Stochastic Langevin Dynamics* [32], which enables the model to approximate complex posterior distributions over actions. As a result, DP demonstrates superior capability in representing both short-horizon and long-horizon multimodal distributions. However, the generated motion trajectories exhibit a lack of efficiency and smoothness due to fundamental architectural constraints in its backbone design. A considerable number of follow-up works have been developed based on DP. For instance, UMI [33] introduces novel data collection methods to enhance diversity, while *DATA SCALING LAWS* [34] leverages DINOv2 [35] fine-tuning and integration strategies to improve temporal coherence. Yet, they all employ a similar fundamental architecture, so the limitation persists. Our work seeks to address the issue through architecture improvements.

III. Methods

To address the discrete feature processing limitations of CNNs and Transformers, this chapter introduces two custom fundamental network architectures, KP-C and KP-T. The overall architecture is shown in Fig. 2.

A. Preliminaries

DDPMs [31] implement a fixed forward diffusion process that progressively corrupts data samples $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ with Gaussian noise through T Markov steps, governed by a predefined variance schedule $\{\beta_k\}_{k=1}^{T}$ where $\beta_k \in (0, 1)$. This yields the closed-form conditional distribution $q(\mathbf{x}_k|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_k; \sqrt{\bar{\alpha}_k}\mathbf{x}_0, (1 - \bar{\alpha}_k)I)$ with $\bar{\alpha}_k = \prod_{i=1}^{k} \alpha_i$ and $\alpha_k = 1 - \beta_k$, I is a Gaussian distribution representing the noise. The reverse process learns parameterized transitions $p_\theta(\mathbf{x}_{k-1}|\mathbf{x}_k) = \mathcal{N}(\mathbf{x}_{k-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_k, k), \sigma_k^2 I)$, initialized from isotropic Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(0, I)$. Here $\boldsymbol{\mu}_\theta$ is implemented as a neural network that predicts the denoising direction, typically reparameterized as $\boldsymbol{\mu}_\theta(\mathbf{x}_k, k) = \frac{1}{\sqrt{\alpha_k}}(\mathbf{x}_k - \frac{\beta_k}{\sqrt{1-\bar{\alpha}_k}}\epsilon_\theta(\mathbf{x}_k, k))$, while the variance σ_k^2 is commonly fixed to $\frac{1-\bar{\alpha}_{k-1}}{1-\bar{\alpha}_k}\beta_k$. Training employs the simplified noise-prediction objective $L_k = \mathbb{E}_{k,\mathbf{x}_0,\boldsymbol{\epsilon}_k}\left[\left\| \epsilon_k - \epsilon_\theta \left(\sqrt{\bar{\alpha}_k}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_k}\epsilon_k, t \right) \right\|^2 \right]$, which directly optimizes the network ϵ_θ to predict the Gaussian noise $\boldsymbol{\epsilon}$ added at each diffusion step k.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

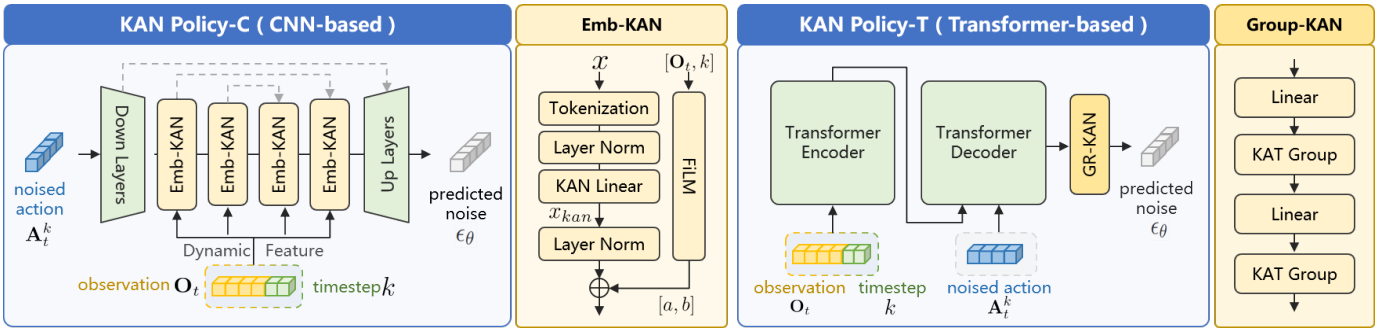

Fig. 2: We present the network architectures of KP, while using CNN (-C) or Transformer (-T) as the backbone network.

Similarly, KP implements a conditional diffusion process grounded in DP [4]. At time step t, the ground truth actions $\mathbf{A}_t^0$ undergoes k-step Gaussian corruption to produce the noised counterpart $\mathbf{A}_t^k$. Correspondingly, the single-step reverse diffusion process conditioned on the observation $\mathbf{O}_t$ can generate the predicted noisy action sequence $\hat{\mathbf{A}}_t^{k-1}$ through the following formulation:

$$\hat{\mathbf{A}}_t^{k-1} = \alpha(\mathbf{A}_t^k - \gamma\epsilon_\theta(\mathbf{O}_t, \mathbf{A}_t^k, k) + \mathcal{N}(0, \sigma^2 I)) \tag{1}$$

Here, α, γ, σ denote diffusion scheduling parameters dependent on denoising step k, and $\mathcal{N}(0, \sigma^2 I)$ represents Gaussian noise. θ represents the learnable network parameter. Executing k denoising steps to produce a sequence of intermediate actions with progressively lower noise levels, culminating in the generation of the noise-free output $\hat{\mathbf{A}}_t^0$. KP is optimized via a modified denoising diffusion objective to estimate the noise ϵ^k added to the input actions for iteration k:

$$\mathcal{L} = \text{MSE}(\epsilon^k, \epsilon_\theta(\mathbf{O}_t, \mathbf{A}_t^0 + \epsilon^k, k)) \tag{2}$$

B. KAN Policy-C

The framework employs a tripartite architecture comprising sequential feature transformation stages (Fig. 2). The Down layers hierarchically condense input signals through convolutional blocks while caching multiscale features via skip connections. At the core of our architecture lies the novel Emb-KANs. It uses a special mathematical approach to adjust interactions between different features, ensuring that the features remain connected. This enhances the network's ability to grasp complex patterns in high-dimensional data. The subsequent Up layers reconstruct target outputs through mirrored convolutional blocks with residual gating mechanisms. They dynamically fuse cached skip features from the Down layers and semantically enriched embeddings from Emb-KAN before the final prediction. This combined approach ensures that features flow coherently and enables the network to adapt its representation to different noise levels. Globally, the Dynamic Feature, generated globally by observation and timestep, is processed by the conditioning encoder Feature wise Linear Modulation (FiLM) [36] in each stage and then incorporated into the current stage's features.

Emb-KAN In this module, Tokenization is designed to convert a one-dimensional sequence into a patch-based embedding representation with overlapping regions. This method effectively extracts local features from sequential data while maintaining continuity between patches. Given an input sequence $x \in R^{B\times C_{in}\times L_{in}}$, where B is the batch size, C_{in} represents the number of input channels, and L_{in} is the sequence length, projected into a higher-dimensional embedding space using a one-dimensional convolution operation:

$$x_{patch} = \text{LN}(\text{Conv}(x, W)) \tag{3}$$

where $W \in R^{C_{out}\times C_{in}\times ks}$ acts as the learnable weights of the operation, ks is the kernel size, and C_{out} is the embedding dimension, LN is the Layer Norm. The convolution operation includes stride s and padding $p = \lfloor ks/2 \rfloor$ to ensure overlapping patches. The output has the shape $x_{patch} \in R^{B\times C_{out}\times L_{patch}}$, where L_{patch} is determined by:

$$L_{patch} = \left\lfloor \frac{L_{in} + 2p - ks}{s} \right\rfloor + 1 \tag{4}$$

Then we pass through a Layer Norm, which processes the features along the embedding dimension.

After the patch-based embedding representation of feature is completed, the primary feature processing within this module is conducted by KAN Linear, whose output x_{kan}, can be mathematically characterized by the following formulation:

$$x_{kan} = w_b \cdot \sigma(x_{patch}) + w_s \cdot B(x_{patch}) \tag{5}$$

In this expression, w_b and w_s denote the weights generated by linear layers in the process. The functions are defined as: $\sigma(x) = x/(1 + e^{-x})$, and $B(x) = \sum_i c_i B_i(x)$ where c_i are the trainable coefficients for the i-th basis function $B_i(x)$. Particularly, when considering a univariate function $B(x)$ defined over the interval $[a, b]$ with B-splines of order k, the interval is divided into G subintervals by $G + 1$ grid points $\{t_0 = a, t_1, \cdots, t_G = b\}$. This set of points is extended to $\{t_{-k}, \cdots, t_{-1}, t_0, t_1, \cdots, t_G, t_{G+1}, \cdots, t_{G+k}\}$ to accommodate the spline order. Following [37], the spline-based function is then defined recursively as:

$$B_{i,k}(x) = \frac{x - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(x) + \frac{t_{i+k} - x}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(x) \tag{6}$$

When $k = 0$, $B_{i,0}(x)$ is defined as 1 if $t_i \le x < t_{i+1}$ and 0 otherwise. This process constructs an interpolation framework within the input feature space, enabling the simultaneous extraction of both linear and nonlinear features.

Finally, the conditioning encoder FiLM is tasked with extracting Dynamic Feature a and b from observation and timestep. So the output $f(x)$ of Emb-KAN is structured as:

$$f(x) = a \cdot \text{LN}(x_{kan}) + b \tag{7}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Push-T **Lift** **Can** **Square** **Transport** **Tool Hang** **Kitchen**

TABLE I: SIMULATION RESULTS ACROSS VARIOUS TASKS

Policy	**Push-T** (ph)				**Lift** (ph)				**Can** (ph)			
	Succ ↑ (%)	Time ↓ (s)	Cur ↓ (m^{-1})	DSJ ↓ (10^{17})	Succ ↑ (%)	Time ↓ (s)	Cur ↓ ($10^3 \cdot m^{-1}$)	DSJ ↓ (10^{16})	Succ ↑ (%)	Time ↓ (s)	Cur ↓ ($10^3 \cdot m^{-1}$)	DSJ ↓ (10^{16})
DP-C	76.7±3.40	17.4±6.44	1.34±1.27	7.03±1.23	**100**±0.00	2.34±0.26	**0.20**±0.04	0.51±0.18	**100**±0.00	5.79±0.87	4.67±0.17	4.34±0.14
KP-C	**84.7**±1.89	**15.8**±5.73	**0.75**±0.56	**4.90**±0.86	**100**±0.00	**2.14**±0.20	0.45±0.19	**0.21**±0.02	**100**±0.00	**5.17**±0.70	**0.80**±0.78	**4.31**±0.32
DP-T	42.7±2.49	21.6±7.34	2.19±0.24	12.4±1.30	**100**±0.00	2.34±0.73	0.97±0.22	0.43±0.05	96.0±4.00	5.86±0.51	1.05±0.13	6.56±1.06
KP-T	**64.0**±1.63	**15.9**±5.68	**1.42**±1.25	**12.4**±1.56	**100**±0.00	**2.16**±0.52	**0.39**±0.09	**0.30**±0.03	**98.7**±2.31	**5.20**±0.55	**0.90**±0.38	**4.01**±0.76
Policy	**Square** (ph)				**Transport** (ph)				**Tool hang** (ph)			
	Succ ↑ (%)	Time ↓ (s)	Cur ↓ ($10^3 \cdot m^{-1}$)	DSJ ↓ (10^{17})	Succ ↑ (%)	Time ↓ (s)	Cur ↓ ($10^3 \cdot m^{-1}$)	DSJ ↓ (10^{20})	Succ ↑ (%)	Time ↓ (s)	Cur ↓ ($10^3 \cdot m^{-1}$)	DSJ ↓ (10^{19})
DP-C	93.3±4.16	7.96±2.06	14.4±1.36	1.07±0.20	90.0±2.00	22.1±2.41	16.5±3.88	7.85±4.79	50.6±8.08	24.8±4.14	**1.82**±0.61	**6.30**±0.20
KP-C	**94.0**±2.00	**7.42**±1.18	**13.0**±0.48	**0.74**±0.07	**94.0**±2.00	**21.6**±2.32	**7.76**±1.46	**7.43**±4.52	**53.3**±7.57	**22.2**±3.13	3.39±1.26	7.36±0.34
DP-T	93.3±1.15	6.87±0.89	**1.01**±0.24	49.0±10.0	86.7±2.31	23.3±3.40	**2.46**±2.37	38.0±5.76	86.7±4.62	23.4±4.10	1.01±0.71	195±43.0
KP-T	**95.3**±2.31	**6.86**±0.94	2.06±1.25	**12.9**±1.67	**89.3**±1.15	**21.6**±2.16	3.25±2.76	**20.8**±4.45	**88.7**±1.15	**21.0**±3.37	**0.58**±0.51	**52.9**±7.92
Policy	**Lift** (mh)				**Can** (mh)				**Square** (mh)			
	Succ ↑ (%)	Time ↓ (s)	Cur ↓ (m^{-1})	DSJ ↓ (10^{16})	Succ ↑ (%)	Time ↓ (s)	Cur ↓ ($10^3 \cdot m^{-1}$)	DSJ ↓ (10^{16})	Succ ↑ (%)	Time ↓ (s)	Cur ↓ ($10^3 \cdot m^{-1}$)	DSJ ↓ (10^{17})
DP-C	**100**±0.00	4.67±2.01	1.95±0.25	0.67±0.15	96.7±1.15	9.08±3.19	24.3±9.95	7.17±1.51	**84.0**±3.46	12.6±3.51	166±22.1	1.06±0.19
KP-C	**100**±0.00	**4.29**±1.36	**1.68**±0.87	**0.15**±0.03	**99.3**±1.15	**8.71**±2.59	**2.69**±1.19	**7.15**±1.46	**84.0**±2.00	**11.6**±4.00	**11.4**±7.56	**0.95**±0.43
DP-T	**100**±0.00	4.19±0.23	**0.58**±0.24	**1.01**±0.05	92.7±6.11	9.61±3.91	1.99±0.21	101±5.62	80.7±3.06	12.6±4.89	13.1±5.18	680±95.2
KP-T	**100**±0.00	**3.92**±0.46	0.73±0.13	3.00±1.01	**98.0**±2.00	**9.23**±2.16	**0.57**±0.26	**26.0**±2.13	**82.0**±5.77	**11.7**±4.03	**6.01**±0.65	**193**±42.3
Policy	**Transport** (mh)				**Kitchen** (ph)							
	Succ ↑ (%)	Time ↓ (s)	Cur ↓ ($10^3 \cdot m^{-1}$)	DSJ ↓ (10^{20})	Succ-1 ↑ (%)	Succ-2 ↑ (%)	Succ-3 ↑ (%)	Succ-4 ↑ (%)	Succ-5 ↑ (%)	Time ↓ (s)	Cur ↓ ($10^3 \cdot m^{-1}$)	DSJ ↓ (10^{16})
DP-C	**62.0**±2.00	26.4±5.00	**17.6**±9.01	38.1±11.6	**100**±0.00	**100**±0.00	**100**±0.00	99.3±0.94	44.0±5.89	23.9±2.40	0.30±0.23	17.0±4.52
KP-C	60.7±4.16	**25.9**±3.67	29.3±16.7	**0.68**±0.03	**100**±0.00	**100**±0.00	**100**±0.00	**100**±0.00	**61.3**±7.54	**21.8**±1.38	**0.25**±0.11	**6.10**±1.89
DP-T	37.3±9.45	30.0±3.68	1.16±0.19	51.4±11.8	**100**±0.00	**100**±0.00	**100**±0.00	**99.3**±0.94	20.7±0.94	28.1±3.11	0.37±0.27	**2.89**±0.77
KP-T	**49.3**±6.43	**27.0**±4.69	**1.04**±0.14	**45.2**±10.4	**100**±0.00	**100**±0.00	**100**±0.00	**99.3**±0.94	**33.3**±11.4	**27.4**±3.27	**0.14**±0.03	7.72±3.09

This comprehensive process extracts and integrates features, collectively capturing the essential characteristics.

C. KAN Policy-T

Our experimental architecture, a transformer-based model for conditional sequence modeling, incorporates KAT Group with rational functions and group-wise parameters [14], differing from conventional spline function usage. As shown in Fig. 2, this versatile model has potential applications in diffusion processes and autoregressive generation. While its overall structure follows [3], we replaced the final linear layer with a custom Group-KAN module. Acting like a traditional activation function, this module introduces non-linearity at the output layer, bridging discrete features and enabling effective learning of continuous representations.

Group-KAN This module can be regarded as a variant of [15], introducing KAT Group at the end of the architecture to enable continuous feature transformations. The formulation $G(x)$ of the entire module can be described as:

$$G(x) = w_2 \cdot K^2(w_1 \cdot K^1(x) + b_1) + b_2 \tag{8}$$

In this formulation, similar to Eq. 5, w_1, w_2 and b_1, b_2 are the weights and bias generated by linear layers, K is the rational function, and the superscript of K indicates the different initialization modes of KAT Group. Assuming i is the index of the current input channel d_{in}, j is the index of the output channel d_{out}, we divide the whole input channels into g groups on average, with $z = \lfloor i \cdot g / d_{in} \rfloor$ being the group number, so that K can be formulated as:

$$K_{i,j}(x) = \sum_{i=1}^{d_{in}} w_{i,j} R_z(x_i) \tag{9}$$

This means that input features from the same group will share parameters. $R(x)$ is the rational function is expressed as:

$$R(x) = \frac{\sum_{i=0}^{m} a_i x^i}{1 + |\sum_{j=1}^{n} b_j x^j|} \tag{10}$$

m and n are the orders of polynomials.

IV. Simulation Experiments

We evaluate KP's performance across various robotic tasks from two benchmarks [1] [2], using state-based policies [4].

A. Simulation Environments and Datasets

Push-T is based on the IBC framework [1]. This task involves moving a T-shaped block to a specified target using a

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

circular end-effector. The initial positions of the block and end-effector are randomized, requiring precise control for accurate block placement through point contacts.

Robomimic is a comprehensive platform for robotic manipulation [2], focusing on imitation learning and offline reinforcement learning. We select five distinct tasks from the platform, each with a skilled human operator's demonstration dataset (Proficient-Human, ph). For four tasks inside (except "Tool Hang"), we have extra data from both skilled and less skilled operators (Multi-Human, mh), giving nine variations total. This dataset can test our method's data requirements.

Kitchen is derived from Relay Policy Learning [38], assesses a policy's ability to handle multiple long-horizon tasks. It comprises 7 interactive objects and mandates the completion of 5 tasks in any order during each demonstration. The goal is to perform as many of the demonstrated tasks as possible, irrespective of their sequence.

B. Evaluation Methodology and Baselines

Criteria. We present *the average success rate* (Succ) of the best-performing checkpoints across three distinct seeds, selecting the top 5 checkpoints for each seed, to ensure robustness in the evaluation. Besides, we report *the mean execution time* (Time) measured in seconds across the intersection of the corresponding best chekpoints' 50 test environments as the ***efficiency*** criterion. To assess trajectory ***smoothness***, we adopt the mean *curvature of trajectories* (Cur) and the mean *dimensionless squared jerk* (DSJ) [39]. Assume $r(t)$ represents the position vector of the trajectory at time $t \in [0, T]$, with $x(t), y(t), z(t)$ denoting the position components in the respective coordinate axes and T is the duration of the generated trajectory. The Cur is determined by:

$$\text{Cur} = \frac{1}{n}\sum_{i=1}^{n}\frac{\left\|r_i^{''}(t)\times r_i^{'}(t)\right\|}{\left\|r_i^{'}(t)\right\|^3+\gamma} \tag{11}$$

Here, $r^{'}(t)$ denotes the velocity vector at time t, $r^{''}(t)$ represents the acceleration vector, γ is an infinitesimal constant. While DSJ in three dimensions can be formulated as:

$$DSJ = \frac{1}{n}\sum_{i=1}^{n}\int_0^{T_i}\left(x_i^{'''}(t)^2 + y_i^{'''}(t)^2 + z_i^{'''}(t)^2\right)dt\frac{T_i^5}{A_i^2} \tag{12}$$

Where parameter A is the total length of the trajectory. i refers to the current environment index, and n represents the number of environments. Additionally, we calculate standard deviations for our primary experimental results to enhance credibility. In some experiments with Push-T, we also use *the coverage area of the target* as a metric, called ***Coverage***. Furthermore, we calculate the average relative improvement using the formula $\frac{1}{n}\times\sum_{i=1}^{n}\frac{(-1)^{op}\times(K_i-D_i)}{D_i}\times 100\%$, where D_i represents the metric of worser, K_i represents the metric being compared, and i denotes the task index, ranging from 1 to n. *op* is set to 0 if higher metric values are desirable, and 1 if lower values are preferred. **The improvement in smoothness is the average of two metircs.**

TABLE II: GENERALIZATION OF GROUP-KAN

Policy	Succ(%) ↑	Time (s)↓	Cur (m^{-1})↓	DSJ (10^{17})↓
BET [3]	34.7	21.6	10e4	75.3
BET+Group-KAN	**43.3**	**12.1**	**3.85**	**0.45**
IBC [1]	50.0	19.8	3.33	0.99
IBC+Group-KAN	**61.3**	**17.8**	**0.63**	**0.37**
DP-C [4]	76.7	17.7	4.92	10.7
DP-C+Group-KAN	**77.3**	**15.6**	**1.08**	**7.54**

TABLE III: ABLATION STUDY BETWEEN MODULES

Policy	Succ (%) ↑	Time (s) ↓	Cur (m^{-1})↓	DSJ (10^{17})↓
DP-C+Group-KAN	77.3	16.2	1.68	35.2
KP-C	**84.7**	**14.3**	**1.38**	**5.57**
DP-T+KAN Linear	46.7	21.7	**0.50**	11.7
KP-T	**64.0**	**18.5**	0.54	**9.61**

Baselines. We conduct a comparison between KP and DP, including their CNN-based (-C) and Transformer-based (-T) variants, on the datasets in Sec. IV-A. To explore the generalization of Group-KAN and the ultimate success rate of tasks, we conduct two additional studies. First, we integrate Group-KAN into IBC [1], BET [3], and DP-C, comparing performance with the original models. Second, we are curious about whether the modules are optimal, so we swap the modules between the two types of networks. Notice that all the additional experiments are conducted on Push-T.

C. Key Findings

KP is capable of generating smooth and efficient trajectories. KP demonstrates superior overall performance, achieving a higher average success rate, shorter execution time, and more smooth trajectory compared to DP. As summarized in Table I, KP outperforms across all evaluation criteria, with average improvements of 6.06%, 8.03%, and 26.4% in success rate, execution time and smoothness spanning diverse datasets. These results collectively validate the effectiveness of our modular design of KANs and their systematic integration into the framework.

KP is robust to variations in datasets. The Multi-Human datasets captured from operators with heterogeneous skill levels—exhibit inherent heterogeneity in motion planning behaviors. Such variability introduces substantial challenges, including: divergence in trajectory geometry (e.g., path length discrepancies), actuation noise (e.g., positioning errors during grasping attempts) and inconsistent task execution patterns. Notably, KP demonstrates enhanced efficacy on these inferior datasets. As quantified in Table I, it achieves statistically significant improvements of 5.01%, 6.20%, and 34.1% in metrics after training in suboptimal datasets, respectively. These findings underscore KP's resilience against dataset variability, outperforming even under suboptimal data conditions—a critical advantage for real-world deployment scenarios where operator expertise varies widely.

While KP has excellent overall performance, KP-C underperforms in "Transport", a dual-robotic-arm cooperation task. We observe that this task has double the "Degree-of-Freedom" (DoF) of other robotic tasks in our experiments with Robotmimic. This performance degradation of KP-C under suboptimal datasets underscores its inherent limitations

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

in high-DoF manipulation tasks, revealing a critical sensitivity to data quality as kinematic complexity scales.

Group-KAN can generalize to more policy learning models. Group-KAN emerges as a universal enhancer for policy learning frameworks. When integrated as the final layer across diverse architectures, it consistently elevates performance metrics, as demonstrated in Table II. Notably, baseline models with initially limited capabilities exhibit the most substantial improvements through this integration. These findings confirm Group-KAN's dual role: 1) as a performance amplifier for advanced strategies like Diffusion Policy, and 2) as a corrective module for suboptimal policies, thereby establishing its versatility in robotic learning systems. The evidence from systematic evaluations supports its generalizability beyond specific architectural constraints.

KP exhibits sustained superiority in complex long-horizon tasks. In the Kitchen task, we further evaluated the performance in completing the fifth task. KP significantly outperformed the baselines, achieving a 50.0% improvement in the success rate for Task 5, a 5.64% enhancement in execution time, and a 20.1% overall increase in smoothness (Table I). This showcases its superior ability to handle complex operational sequences. Furthermore, we found that KP demonstrated superior performance in experiments with an extended action horizon, further validating our claim.

We employed the most effective integration method for our approach. We conducted a comparative study on the two KAN - based modules we developed, interchanging the two network types combined with the modules. (Since Emb-KAN is specifically tailored for CNN-based types, we used KAN Linear for the combination rather than the entire module.) The results are presented in Table III. This architecture space exploration empirically validates our principled selection of the optimal model integration strategy.

V. Real-World Tasks

To evaluate KP's real-world performance, we designed and collected three tasks, namely Push Cup, Sort and Give. The baselines and evaluation metrics for real-world tasks were consistent with those used in simulation experiments.

A. Hardware Tools and Data Preparation

As illustrated in Fig. 4(a), our experimental setup features an 6-DoF robotic arm named RM65-6F integrated with a 3D-printed Robotic Gripper serving as the end-effector. The robotic system incorporates a Wrist Depth Camera directly integrated into the gripper assembly, capable of capturing 4K-resolution video streams for real-time visual feedback enabled by high-bandwidth data transmission. For data acquisition, the toolkit is simply the Robotic Gripper with camera.

We gather our dataset using UMI's handheld interface [33]. Specially, UMI employs a pre-trained vision encoder [40] to extract visual representations from real-world human demonstrations, which are then used to train a visuomotor policy that can generate implementable robot control actions. Data acquisition follows a structured protocol that involves recording the task-execution process. Following data acquisition, we processed the data across multiple environments to enhance the generalization capability of the policy [34]. Datasets were structured to include synchronized observations of such as RGB images and gripper width extracted through SLAM and IMU data fusion. This approach ensure that the policy can learn from diverse environmental conditions and objects, improving its robustness and adaptability. Policy training and validation were executed on dedicated computing servers.

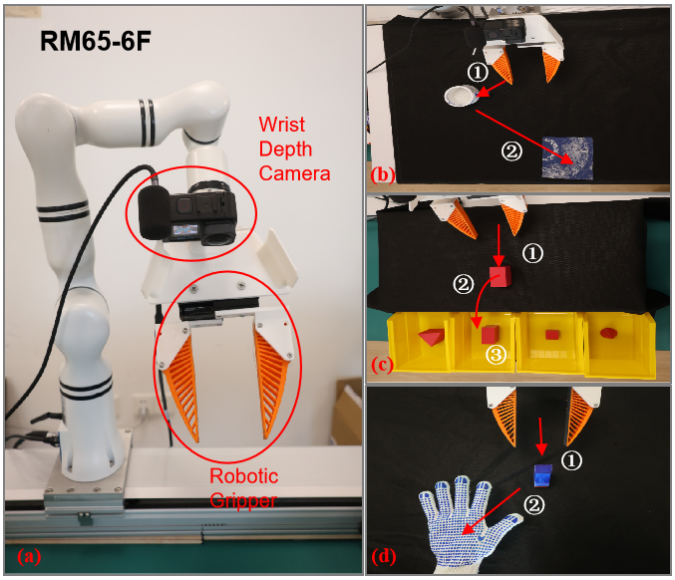

Fig. 3: Real-World experimental devices and settings. (a) Specifications of the robotic arm device. (b) Push Cup: The robot is required to first ① locate the cup and then ② push it entirely into the blue area. (c) Sort: The robot must first ① pick up the object, then ② identify the box holding the object of the same shape, and finally ③ place it inside the box. (d) Give: The robot need to ① grasp the object and ② transport it to a randomly target.

B. Push Cup

Settings. The task, corresponding to Push-T [1] in simulation environment but is more challenging in the real world due to the physical shape of the gripper, was conducted primarily to evaluate the trajectory planning capabilities of the relevant methods. The task will be deemed successful when the entire bottom of the cup is contained within the blue area. A failure occurs if the bottom of the cup either presses against or crosses the boundary line of the blue area.

Implementation. In total, we collected 410 videos for training, covering three directions. A total of 90 tests were conducted, comprising 30 tests in each of the three directions.

Result Analysis. The experimental results are presented in Table IV. KP improves success rates by 38.9% and 15.6%, and reduces average times by 1.11 and 2.56 seconds for the CNN-based and Transformer-based models respectively. KP-C maintains a stable curvature despite certain variations, while KP-T achieves a better smoothness metric than DP-T. Furthermore, most DP failures are due to faulty trajectory planning, with cups either missing the target area entirely (75% of failures) or not being fully pushed into it. KP failures mainly occur when the cup is only partially pushed into the target area. Based on these results, our method demonstrates superior trajectory planning capabilities through higher success rates, faster completion times, and stable smoothness.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE IV: REAL-WORLD EXPERIMENT RESULTS

Policy	Push Cup				Sort				Give			
	Succ↑ (%)	Time↓ (s)	Cur↓ ($10^4 \cdot$ m^{-1})	DSJ↓ (10^{16})	Succ↑ (%)	Time↓ (s)	Cur↓ ($10^4 \cdot$ m^{-1})	DSJ↓ (10^{17})	Succ↑ (%)	Time↓ (s)	Cur↓ ($10^4 \cdot$ m^{-1})	DSJ↓ (10^{16})
DP-C	44.4	11.5$_{\pm 3.42}$	**3.21**$_{\pm 0.72}$	27.9$_{\pm 8.48}$	20.0	15.5$_{\pm 2.72}$	6.02$_{\pm 0.77}$	44.9$_{\pm 11.1}$	50.0	18.1$_{\pm 6.07}$	5.48$_{\pm 2.76}$	6.27$_{\pm 1.01}$
KP-C	**83.3**	**10.4**$_{\pm 3.98}$	3.47$_{\pm 0.55}$	**4.50**$_{\pm 1.40}$	**35.0**	**12.3**$_{\pm 3.63}$	**5.21**$_{\pm 0.81}$	**0.24**$_{\pm 0.05}$	**80.0**	**14.7**$_{\pm 9.13}$	**3.43**$_{\pm 0.27}$	**5.14**$_{\pm 1.61}$
DP-T	52.2	12.0$_{\pm 6.82}$	9.51$_{\pm 2.65}$	**2.52**$_{\pm 0.76}$	10.0	**9.50**$_{\pm 1.73}$	5.66$_{\pm 0.63}$	**0.54**$_{\pm 0.09}$	50.0	18.3$_{\pm 6.63}$	4.37$_{\pm 0.40}$	1.99$_{\pm 0.51}$
KP-T	**67.8**	**9.50**$_{\pm 3.98}$	**3.94**$_{\pm 0.55}$	3.24$_{\pm 2.51}$	**15.0**	13.6$_{\pm 3.24}$	**3.05**$_{\pm 0.24}$	1.49$_{\pm 0.15}$	**60.0**	**16.9**$_{\pm 3.10}$	**4.18**$_{\pm 0.34}$	**0.30**$_{\pm 0.10}$

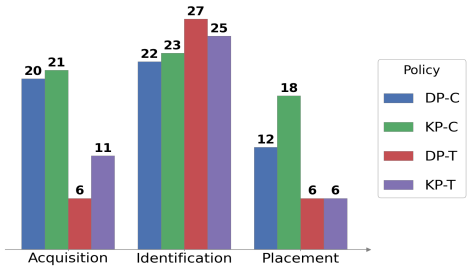

Fig. 4: For further analysis of Sort, we recorded the success count per-step.

C. Sort

Settings. This task draws inspiration from the 'Can' task in virtual environments, involving placing a fixed-shape object into a corresponding box. The challenge lies in identifying the box with the same-shape object, as all objects have identical colors and similar sizes, and their shapes may also be close from certain viewpoints.

Implementation. We collected 600 videos similarly with IMU data from three distinct environmental scenarios, with each collection focusing solely on cuboid (shapes encompassing cuboid, cylinder, cone, and cube). Each box underwent ten tests, resulting in a total of 40 tests per model.

Result Analysis. Compared to DP-C and DP-T, KP-C and KP-T increase success rates by 15% and 5%, respectively, and produce smoother outcomes. The success rate is a more important performance metric in this precision-dependent task, as achieving the core manipulation objective is more critical. KP's better performance indicates our method's capability to handle more difficult tasks in a certain case.

However, this task achieved a lower success rate. To investigate the reasons, we conducted further analysis to determine which step is most prone to failure (Fig. 4). We define three key steps: ① Acquisition (pick the object), ② Identification (find the box, occur independently), ③ Placement (place object inside box). For DP-C and KP-C, task success hinges on Placement if Acquisition is successful, as Identification failures are relatively rare but sometimes Placement don't occur, which is manifested as gripping the object tightly without releasing. The probability of successful Acquisition is nearly 50%. Once Acquisition is achieved, KP's overall task success rate approaches 70%. For DP-T and KP-T, task success hinges on both Acquisition and Placement, as Identification failures are less common. The probability of successful Acquisition here is only 25%. Similarly, once Acquisition is successful, the overall task success rate is approximately 55%. We believe that the low success rate of Acquisition is the primary reason for the overall low success rate because most failures are directly or indirectly due to Acquisition issues. With consistent experimental conditions, our conclusions still remain valid.

D. Give

Settings. This dynamic human-robot interaction task requires transferring a gripper component onto a randomly positioned hand embedded in a specialized glove. To simulate dynamic environmental conditions, additional glove-mounted receptacles are introduced as distractors.

Implementation. A dataset of almost 140 demonstration videos was compiled for model training, supplemented by 20 independent evaluation trials to assess system performance. The trials is consistent for all policies to fairly measure.

Result Analysis. In the Give task, KP delivered an extraordinary performance, achieving an average 20% increase in success rate. It also maintained superior performance in terms of efficiency and smoothness. We observed that KP rapidly track and lock onto the correct dynamic target for successful delivery, with failures exclusively occurring due to the grasped object slipping from the hand. While DP employs improper grasping postures, leading to collision-induced failures. This demonstrates the excellent performance of our method in dynamic tasks and also verifies the advantages of KP in efficiency and smoothness.

E. Discussion

During training, we found that KP-C adds about 4% parameters, boosting average success rate by 28.0%, while KP-T has almost no parameter increase and improves average success rate by 10.2%. Considering training time, with Push Cup task, DP-C takes about 67.1 average wall-clock hours, KP-C about 68.0; DP-T takes about 63.1, and KP-T about 63.4. Thus, adding KAN modules doesn't significantly increase training time, indicating that using KAN modules has relatively low practical costs. In practical deployment, our experiments typically run at 10 Hz, adjustable per task. For instance, the Give task employ 20 Hz control for real-time deployment. KP can operate within these frequency requirements without causing significant delays. KP maintains timely responses for Give task, which indicates that it meets the max requirements for our real-time deployment.

VI. Conclusions

In this work, we introduce the *KAN Policy* (KP) to address the limitations of existing Diffusion Policy (DP) models in generating efficient and smooth trajectories. Specifically, we designed a novel *Embedding KAN* (Emb-KAN) module for CNN-based DP models to provide continuous structural representations in high-dimensional embedding spaces. Additionally, we propose a novel pipeline integrating KANs to Transformer-based DP models, thereby expanding the applicability of KP across different architectures. Extensive

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

experiments across diverse simulation and real-world tasks demonstrate significant improvements on success rate, execution time, and smoothness of KP over baseline methods.

REFERENCES

[1] P. Florence, C. Lynch, A. Zeng, O. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conf. Robot. Learn. (CoRL)*, 2021, pp. 488–501.

[2] 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," in *Conf. Robot. Learn. (CoRL)*, 2021, pp. 1336–1352.

[3] N. M. M. Shafiullah, Z. J. Cui, A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," 2022, arXiv:2206.11251.

[4] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proc. Robot. Sci. Syst. (RSS)*, 2023.

[5] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3D diffusion policy," 2024, arXiv:2403.03954.

[6] B. Kang, X. Ma, C. Du, T. Pang, and S. Yan, "Efficient diffusion policies for offline reinforcement learning," in *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2024.

[7] X. Ma, S. Patidar, I. Haughton, and S. James, "Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2024, pp. 18 081–18 090.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Int. Conf. Learn. Represent. (ICLR)*, 2021.

[10] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, *Path Planning and Trajectory Planning Algorithms: A General Overview*. Springer, 2015, pp. 3–27.

[11] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov-Arnold networks," 2024, arXiv:2404.19756.

[12] 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," 2024, arXiv:2406.02918.

[13] N. Firsov, E. Myasnikov, V. Lobanov, R. Khabibullin, N. Kazanskiy, S. Khonina, M. A. Butt, and A. Nikonorov, "HyperKAN: Kolmogorov–Arnold networks make hyperspectral image classifiers smarter," *Sensors*, vol. 24, no. 23, p. 7683, 2024.

[14] X. Yang and X. Wang, "Kolmogorov-Arnold transformer," 2024, arXiv:2409.10594.

[15] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[16] M. Unser, A. Aldroubi, and M. Eden, "B-spline signal processing. Part I. Theory," *IEEE Trans. Signal Process.*, vol. 41, no. 2, pp. 821–833, Feb. 1993.

[17] W. Aribowo and K. Terashima, "Cubic spline trajectory planning and vibration suppression of semiconductor wafer transfer robot arm," *Int. J. Autom. Technol.*, vol. 8, no. 2, pp. 265–274, 2014.

[18] G. Braglia, M. Tagliavini, F. Pini, and L. Biagiotti, "Online motion planning for safe human–robot cooperation using B-Splines and hidden Markov models," *Robotics*, vol. 12, no. 4, p. 118, 2023.

[19] H. Lurz, T. Recker, and A. Raatz, "Spline-based path planning and reconfiguration for rigid multi-robot formations," vol. 106, 2022, pp. 174–179, 9th CIRP Conference on Assembly Technology and Systems.

[20] A. N. Kolmogorov, "On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition," *Dokl. Akad. Nauk SSSR*, vol. 114, pp. 953–956, 1957.

[21] V. I. Arnol, "On the representation of continuous functions of three variables by superpositions of continuous functions of two variables," *Dokl. Akad. Nauk SSSR*, vol. 114, pp. 679–681, 1957.

[22] J. Braun and M. Griebel, "On a constructive proof of Kolmogorov's superposition theorem," *Constr. Approx.*, vol. 30, pp. 653–675, 2009.

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent. (MICCAI)*, 2015, pp. 234–241.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017, pp. 5998–6008.

[25] H. Leung and S. Haykin, "Rational function neural network," *Neural Comput.*, vol. 5, no. 6, pp. 928–938, 1993.

[26] M. Telgarsky, "Neural networks and rational functions," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 3343–3351.

[27] N. Boullé, Y. Nakatsukasa, and A. Townsend, "Rational neural networks," in *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020, pp. 14 344–14 353.

[28] V. A. Kich, J. A. Bottega, R. Steinmetz, R. B. Grando, A. Yorozu, and A. Ohya, "Kolmogorov-Arnold network for online reinforcement learning," in *Proc. Int. Conf. Control Autom. Syst. (ICCAS)*, 2024, pp. 958–963.

[29] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," in *Advances in Neural Information Processing Systems 27 (NIPS)*, 2014, pp. 2672–2680.

[30] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Int. Conf. Learn. Represent. (ICLR)*, 2014.

[31] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020, pp. 6528–6539.

[32] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 681–688.

[33] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song, "Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots," in *Proc. Robot. Sci. Syst. (RSS)*, 2024.

[34] F. Lin, Y. Hu, P. Sheng, C. Wen, J. You, and Y. Gao, "Data scaling laws in imitation learning for robotic manipulation," 2024, arXiv:2410.18647.

[35] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, H. Jégou, P. Gu, A. Joulin, and P. Bojanowski, "DINOv2: Learning robust visual features without supervision," in *Int. Conf. Learn. Represent. (ICLR)*, 2024.

[36] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018.

[37] C. de Boor, "On calculating with B-splines," *J. Approx. Theory*, vol. 6, no. 1, pp. 50–62, 1972.

[38] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," in *Conf. Robot. Learn. (CoRL)*, 2019, pp. 938–953.

[39] H. Chen, W. Xu, W. Guo, and X. Sheng, "Variable admittance control using velocity-curvature patterns to enhance physical human-robot interaction," *IEEE Robot. Autom. Lett.*, June 2024.

[40] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proc. 38th Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 8748–8763.