

# Hybrid Diffusion Policies with Projective Geometric Algebra for Efficient Robot Manipulation Learning

Xiatao Sun<sup>1</sup>, Yuxuan Wang<sup>2</sup>, Shuo Yang<sup>3</sup>, Yinxing Chen<sup>1</sup>, Daniel Rakita<sup>1</sup>

**Abstract**—Diffusion policies are a powerful paradigm for robot learning, but their training is often inefficient. A key reason is that networks must relearn fundamental spatial concepts, such as translations and rotations, from scratch for every new task. To alleviate this redundancy, we propose embedding geometric inductive biases directly into the network architecture using Projective Geometric Algebra (PGA). PGA provides a unified algebraic framework for representing geometric primitives and transformations, allowing neural networks to reason about spatial structure more effectively. In this paper, we introduce hPGA-DP, a novel hybrid diffusion policy that capitalizes on these benefits. Our architecture leverages the Projective Geometric Algebra Transformer (P-GATr) as a state encoder and action decoder, while employing established U-Net or Transformer-based modules for the core denoising process. Through extensive experiments and ablation studies in both simulated and real-world environments, we demonstrate that hPGA-DP significantly improves task performance and training efficiency. Notably, our hybrid approach achieves substantially faster convergence compared to both standard diffusion policies and architectures that rely solely on P-GATr. The project website is available at: <https://apollo-lab-yale.github.io/26-ICRA-hPGA-website/>.

## I. INTRODUCTION

Diffusion policies [6] have emerged as a powerful paradigm for visuomotor control in robotics, offering reliable convergence through iterative denoising of action trajectories. These models are typically trained from scratch with hundreds of epochs, and while effective, this training paradigm presents a critical inefficiency: the network must repeatedly relearn basic spatial priors—such as translations and rotations—for every new task or environment. This redundant relearning not only inflates the computational cost but also slows down convergence.

Given that spatial concepts are inherently universal across robotic tasks, integrating geometric inductive biases directly into the network architecture presents a compelling strategy to alleviate this redundancy. Projective Geometric Algebra (PGA), a mathematical framework offering a unified representation of spatial entities and operations through mathematical objects called *multivectors*, is particularly suited to

embedding such biases. Multivectors provide a structured algebraic and geometric representation for spatial priors like points, translations, and rotations, which enables neural networks to perform spatial computations more efficiently and intuitively, thereby potentially improving training efficiency and task performance. For a thorough and practical treatment of PGA, we refer the reader to the work by Dorst and De Keninck [7].

To incorporate PGA into general learning tasks, prior research has proposed the Projective Geometric Algebra Transformer (P-GATr) [3], demonstrating its effectiveness in a set of initial spatial learning tasks, such as simulated n-body modeling and predicting shear stress in human arteries, outperforming non-geometric architectures. However, incorporating P-GATr directly as a denoising backbone within diffusion policies is challenging in robotics contexts. As our experiments suggest, the inherent geometric inductive biases and the complexity of multivector computations within P-GATr make it difficult to directly learn noise prediction for effective denoising, resulting in prohibitively slow convergence.

To overcome these limitations, we propose the **hybrid Projective Geometric Algebra Diffusion Policy (hPGA-DP)**, a hybrid diffusion policy architecture that predicts an action sequence based on an observation sequence of the concatenations of the proprioceptive states of the robot and poses of task-relevant objects. The approach is designed to leverage the strengths of both geometric and traditional neural network approaches. Specifically, we employ P-GATr as a spatial state encoder and action decoder, while utilizing established architectures such as U-Net [29] or Transformer [39] for the denoising module. This hybrid approach allows the strong geometric inductive biases of P-GATr to efficiently embed spatial structures into a representation space amenable to effective denoising, while simultaneously benefiting from the proven denoising capabilities of conventional architectures. To our knowledge, our work is the first to incorporate PGA into network architecture for diffusion policies.

We evaluate and validate hPGA-DP through several experiments and ablation studies in both simulated and real-world scenarios. Our results demonstrate that hPGA-DP successfully addresses the convergence issues observed when solely relying on P-GATr, achieving significantly faster convergence and superior task performance compared to standard Transformer or U-Net-based diffusion policy architectures.

We open-source our code to facilitate future research on

<sup>1</sup>Xiatao Sun, Yinxing Chen, Daniel Rakita are with the Department of Computer Science, Yale University, New Haven, CT 06520, USA {xiatao.sun, j.y.chen, daniel.rakita}@yale.edu

<sup>2</sup>Yuxuan Wang is with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, USA wangy49@seas.upenn.edu

<sup>3</sup>Shuo Yang is with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA yangs1@seas.upenn.edu

This work was supported by Office of Naval Research award N00014-24-1-2124

diffusion policies with geometric algebra.<sup>1</sup>

## II. RELATED WORKS

### A. Diffusion Policy

Diffusion models, originally proposed by Ho et al. [13], generate data by reversing a stochastic forward process, and have achieved remarkable success in image and video synthesis [25, 10]. These models have since been adapted for robotic motion generation, with diffusion policies becoming a prominent paradigm for robot learning [6, 40], albeit requiring extensive training over hundreds of epochs.

To mitigate this issue, several works aim to improve training or inference efficiency: Ze et al. [46] proposed using point clouds for richer input; Wang et al. [41] imposed symmetry in denoising to boost generalization; Sun et al. [37] exploited low-rank overparameterization to accelerate training; and Reuss et al. [27] employed mixture-of-experts to reduce inference cost.

While effective, these efforts largely target data efficiency, modality, or inference, without altering the learnable network backbone. Prior architectural innovations centered on Transformers [14, 42], yielding only modest improvements. In contrast, our hPGA-DP introduces a novel hybrid architecture that achieves substantially better training efficiency and policy performance.

### B. Geometric Algebra for Robot Learning

Geometric Algebra (GA) offers a unified framework for representing geometric entities and transformations [8]. Among its variants, Projective Geometric Algebra (PGA) is tailored for Euclidean geometry, using planes as primitives to represent motions compactly with just four basis elements [7, 28]. While GA is mathematically mature, its adoption in robotics and machine learning remains limited, with most applications favoring Conformal Geometric Algebra (CGA) [49, 5] due to its symbolic compactness, as seen in manipulation control [20] and tactile ergodic control [2].

For robotics tasks grounded in Euclidean spaces, PGA ( $\mathbb{G}_{3,0,1}$ ) offers a computationally simpler yet expressive alternative [3, 30]. Existing PGA applications have largely focused on dynamics modeling [33, 34], with limited exploration in learning due to the scarcity of GA-compatible neural architectures [18, 47]. The recent Projective Geometric Algebra Transformer (P-GATr) [3] addresses this gap by embedding geometric inductive biases for geometric learning [4, 17]. Unlike the original P-GATr work, which applied a model-based diffuser via reinforcement learning [15] only in numerical simulations, we introduce the first integration of P-GATr into a diffusion policy via imitation learning [6], validating it on both complex physical simulations and real-world tasks.

In robotics, a concurrent work [48] applied P-GATr within diffusion models for grasp generation, but the approach is limited to static tasks and requires long training times. In contrast, our proposed hPGA-DP is an end-to-end diffusion

policy architecture that achieves superior performance and faster convergence compared to baseline methods across a broad range of robotic learning tasks.

## III. TECHNICAL OVERVIEW

In this section, we present an overview of our proposed hPGA-DP approach. The goal of hPGA-DP is to learn a receding horizon policy  $\pi_\theta$  that outputs a sequence of actions  $\mathbf{a}^{t:t+H_p}$  conditioned on a sequence of observations  $\mathbf{o}^{t-H_o:t}$ , where  $H_p$  and  $H_o$  denote the action prediction and observation horizons, respectively.

### A. Observations

Each observation at time  $t$  is defined as  $\mathbf{o}_t = [\mathbf{s}_t, \{\mathbf{T}_i^t\}_{i=1}^J]$ , where  $\mathbf{s}_t \in \mathbb{R}^{d_s}$  represents the robot state, and  $\{\mathbf{T}_i^t\}_{i=1}^J$  denotes the spatial poses of the  $J$  task-relevant objects. Each pose  $\mathbf{T}_i^t$  consists of a 3D position vector and a unit quaternion representing orientation.

### B. Actions

Each action at time  $\mathbf{a}_t$  may include spatial positions and/or orientations for key links on a robot or any other scalar properties that can affect change on the robot platform. For instance, an action may include the end-effector position at the given time,  $\mathbf{p}_{ee}^t \in \mathbb{R}^3$ , the end-effector orientation at the given time,  $\mathbf{q}_{ee}^t \in \mathbb{H}_1$  (the space of unit-quaternions), and scalar representing how open or closed the gripper is at the given time  $g^t \in \mathbb{R}$ . The action representation is flexible and can include any number platform-specific properties, accommodating various embodiments such as bimanual or humanoid robots with multiple end-effectors. We assume that any system adopting our approach can convert these high-level actions into corresponding robot states or control commands, such as through an inverse kinematics solver.

### C. Architecture

The network architecture for hPGA-DP is illustrated in Fig. 1. First, the robot states from the input observation sequence are converted into the positions and orientations of key links, such as end-effectors, using the robot’s forward kinematics model, in a manner similar to the action representation described above. These spatial components, combined with the poses of task-relevant objects, are then transformed into *multivectors*, the central representational objects in geometric algebra. For the detailed conversions to multivectors, we refer the reader to the practical treatment by Dorst and De Keninck [7].

The resulting multivectors are stacked in a tensor  $\mathbf{x}_o^{t-H_o:t} \in \mathbb{R}^{H_o \times K_o \times 16}$ , where  $H_o$  is the observation horizon,  $K_o$  is the total number positions and orientations of key links on the robot and task-relevant objects in the observations, and 16 is the length of a multivector in  $\mathbb{G}_{3,0,1}$ . The multivector stack is then processed by a P-GATr state encoder [3] to produce an observation latent,  $\mathbf{z}_o \in \mathbb{R}^{H_o \times K_o \times 16}$ . Note that this latent representation maintains the same dimensionality as the input multivector stack.

<sup>1</sup><https://github.com/Apollo-Lab-Yale/hpga-dp>

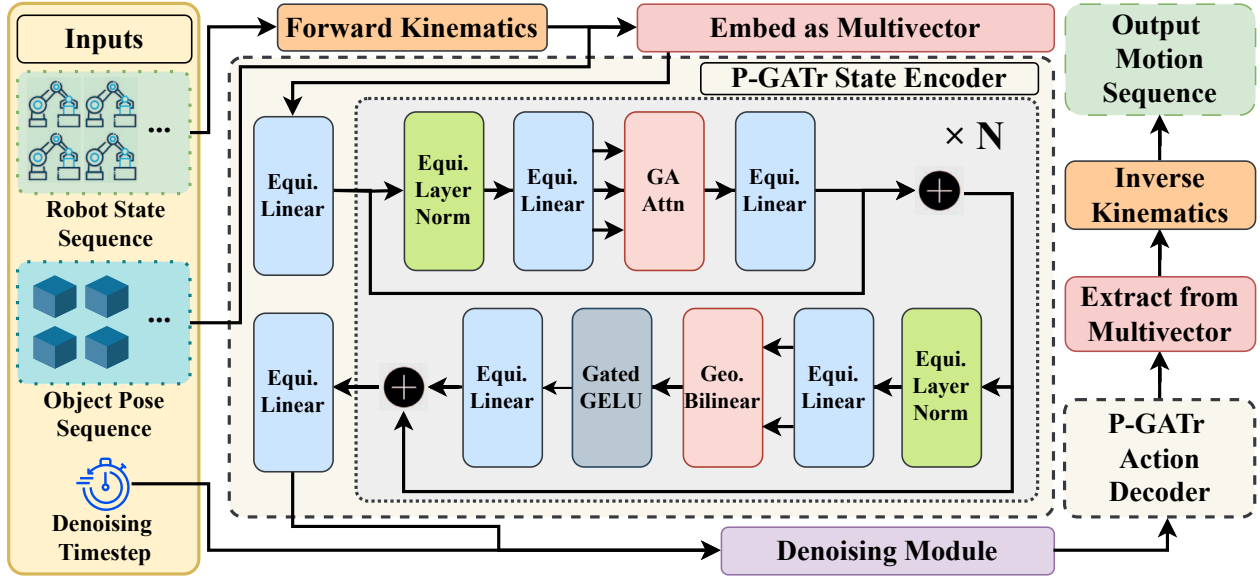


Fig. 1: Overview of the hPGA-DP network architecture.

The observation latent tensor  $\mathbf{z}_o$  is passed to a denoising module, implemented as either a Transformer or U-Net, which produces denoised action latents  $\mathbf{z}_a \in \mathbb{R}^{H_p \times K_a \times 16}$ . Here,  $H_p$  denotes the prediction horizon,  $K_a$  is the total number of positional, rotational, or scalar components in an action, and 16 is the dimensionality of a multivector in  $\mathbb{G}_{3,0,1}$ . These action latents are then decoded by a P-GATr action decoder [3], which mirrors the structure of the state encoder.

The decoder produces a stack of action multivectors as a tensor  $\mathbf{x}_a^{t:t+H_p} \in \mathbb{R}^{H_p \times K_a \times 16}$ , where each multivector represents a predicted position, orientation, or scalar action component. This tensor is unpacked into individual multivectors corresponding to specific link or scalar properties at each time step, and then converted into standard geometric representations such as 3D positions, unit quaternions, and scalar values (e.g., for gripper control). The resulting outputs can be passed to a controller or inverse kinematics solver to recover the final sequence of robot states.

A more in-depth explanation and design justification for this architecture is presented in §IV.

#### D. Projective Geometric Algebra Transformer (P-GATr)

For both the state encoder and action decoder in our architecture, we employ the P-GATr model and associated network primitives introduced by Brehmer et al. [3]. Structurally, P-GATr resembles a standard Transformer [39] with pre-layer normalization [45, 1]. Each of the  $N$  Transformer blocks in P-GATr includes an Equivariant Linear layer (Equi. Linear), Geometric Bilinear layers (Geo. Bilinear), Multivector Attention (GA Attn), a Scalar-Gated Gaussian Error Linear Unit (Gated GELU) [11], and an E(3)-Equivariant LayerNorm (Equi. LayerNorm). A detailed description of these components can be found in the original P-GATr paper by Brehmer et al. [3].

#### IV. ARCHITECTURE DESIGN CHOICES AND DETAILS

In this section, we describe the network design choices and training procedure of hPGA-DP. In preliminary experiments, we initially employed P-GATr directly as the denoising network, aiming to leverage its strong geometric inductive bias. However, we observed that this naive integration resulted in impractically slow convergence, consistent with results in concurrent work by Zhong and Allen-Blanchette [48], which reports week-long training times. We hypothesize that this inefficiency stems from a fundamental mismatch in inductive priors: P-GATr is tailored for processing structured geometric data, whereas the objective of the denoising module is to reverse a stochastic process.

Based on this observation, we design hPGA-DP such that traditional architectures, such as Transformers or U-Nets, serve as the denoising backbone, while P-GATr is utilized as an encoder for observations and a decoder for actions. This allows the denoising process to operate in a latent space where geometric structure is learned implicitly through P-GATr, but without constraining the denoising network itself to a deterministic geometric bias.

Following standard diffusion frameworks [6, 13], we apply the forward noising process to the action latent space:

$$\mathbf{z}_{a,k} = \sqrt{\bar{\alpha}_k} \mathbf{z}_{a,0} + \sqrt{1 - \bar{\alpha}_k} \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I}),$$

where  $\mathbf{z}_{a,k}$  denotes the noisy latent at denoising step  $k$ ,  $\mathbf{z}_{a,0}$  is the clean latent representation of the action sequence,  $\bar{\alpha}_k$  is the cumulative signal retention factor from the noise scheduler that monotonically decreases as  $k$  increases, and  $\boldsymbol{\varepsilon}$  is sampled from a standard Gaussian.

The state encoder and denoising module  $\epsilon_\theta$  are trained to predict the added noise, with the following mean squared error loss:

$$\mathcal{L}_{\text{Encode\&Denoise}} = \|\epsilon_\theta(\mathbf{z}_{a,k}, \mathbf{z}_o, k) - \boldsymbol{\varepsilon}\|^2.$$

where  $\epsilon_\theta$  takes as input the noisy latent  $\mathbf{z}_{\mathbf{a},k}$  at timestep  $k$ , and is conditioned on the observation latent  $\mathbf{z}_{\mathbf{o}}$  to predict the noise  $\epsilon$ .

Although the state encoder and denoising module are jointly trained using the loss  $\mathcal{L}_{\text{Encode\&Denoise}}$ , the action decoder is intentionally excluded from this objective. This design choice stems from the fact that diffusion models are trained by sampling random denoising steps, where the model predicts noise at arbitrary points in the denoising trajectory in parallel. In contrast, inference proceeds sequentially through all denoising steps in an iterative manner. If the action decoder were used across all training steps, it would become entangled again with the denoising process, requiring it to decode from highly noisy action latents that are not suitable for the geometric inductive bias of P-GATr.

To address this problem, we restrict the decoder’s supervision to the final  $\eta$  fraction of denoising steps, where  $\eta$  is a threshold percentage to mask the loss for decoder. The threshold denoising step is calculated by:

$$K_{\text{thresh}} = K_{\text{max}} - \lfloor \eta \cdot K_{\text{max}} \rfloor,$$

where  $K_{\text{max}}$  is the total number of denoising steps and  $\eta \in [0, 1]$  is a tunable parameter. We empirically show that this approach is robust to the choice of  $\eta$  in ablation studies. By limiting supervision to well-denoised action latents, the decoder learns to operate in a structured regime closer to what it encounters during inference, avoiding the need to decode from pure noise.

Since the denoising module  $\epsilon_\theta$  predicts the noise added to the clean latent, we first use its predicted noise  $\hat{\epsilon}$  to compute the estimated denoised action latent  $\hat{\mathbf{z}}_{\mathbf{a},0}$  via the standard reverse process from Ho et al. [13]:

$$\hat{\mathbf{z}}_{\mathbf{a},0} = \frac{1}{\sqrt{\bar{\alpha}_k}} (\mathbf{z}_{\mathbf{a},k} - \sqrt{1 - \bar{\alpha}_k} \cdot \hat{\epsilon}),$$

where  $\bar{\alpha}_k$  is the cumulative signal retention factor at timestep  $k$ , which we also use for the forward noising process.

The decoder is then trained to reconstruct the ground-truth action multivector sequence  $\mathbf{x}_{\mathbf{a}}^{t:t+H_p}$  from  $\hat{\mathbf{z}}_{\mathbf{a},0}$ . The decoder loss is defined as:

$$\mathcal{L}_{\text{Decoder}} = \mathbf{1}_{\{k \geq K_{\text{thresh}}\}} \cdot \left\| D_\phi(\hat{\mathbf{z}}_{\mathbf{a},0}) - \mathbf{x}_{\mathbf{a}}^{t:t+H_p} \right\|^2,$$

where  $D_\phi$  denotes the action decoder, and  $\mathbf{1}_{\{\cdot\}}$  is an indicator function that activates only when the current denoising step  $k$  exceeds the threshold  $K_{\text{thresh}}$ .

This staged supervision strategy preserves architectural modularity, leverages geometric inductive biases of PGA, and speeds up training by limiting the decoder’s learning signal to geometrically meaningful latents. The overall training objective of hPGA-DP combines both loss components:

$$\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{Encode\&Denoise}} + \mathcal{L}_{\text{Decoder}}.$$

## V. EVALUATION

### A. Simulation Experiments

1) *Experimental Settings:* We evaluate the proposed hPGA-DP framework through simulation experiments and

ablation studies across five Robosuite tasks [50], using a 7-DOF Panda Arm. As shown in the top row of Fig. 2, the tasks include: Lift (lift a red cube), Can (sort a can), Stack (stack a red cube on a green one), Square (insert a square nut), and Mug (place a mug into a drawer). Demonstrations for Lift and Can are sourced from Robomimic [21], while those for Stack, Square, and Mug are generated using MimicGen [22]. Each dataset contains 200 trajectories, except Mug, which uses 300 due to its higher complexity.

We compare two hPGA-DP variants: hPGA-U and hPGA-T, which use U-Net and Transformer denoising modules, respectively. Given that our focus is on architectural innovation, we compare against U-Net and Transformer as they are the most widely used backbones for diffusion policies, in addition to a standalone P-GATr backbones without the encoder-denoiser-decoder structure. The U-Net and Transformer baselines contain 24M and 35M parameters, which are the same as the respective number of parameters in hPGA-U and hPGA-T. For object pose input, we test both ground-truth state access (State) and vision-based input via 6D pose estimation using PRISM-DP [36] with FoundationPose [43], leveraging RGB, depth, and generated mesh inputs.

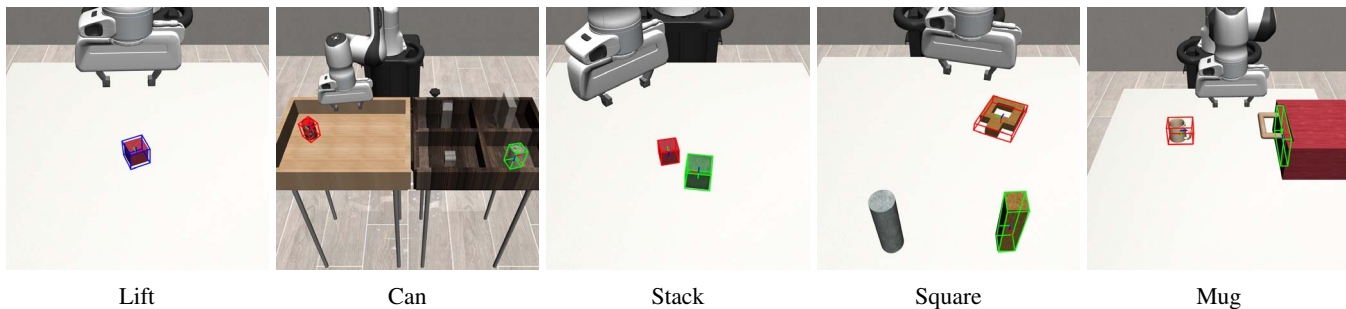
All models are implemented in PyTorch [24] and trained on a workstation with an AMD PRO 5975WX CPU, dual NVIDIA RTX 4090 GPUs, and 128GB RAM. Training schedules are: 80 epochs (Lift), 90 (Can), 30 (Stack), 120 (Square), and 100 (Mug). For all hPGA-DP variants, the maximum denoising step is set to  $K_{\text{max}} = 100$ , and the action decoder loss is applied only during the final  $\eta = 0.25$  portion of the denoising steps.

Each policy is evaluated over 50 rollouts, and success rate, which is defined as the fraction of successful trials, is reported as the primary performance metric, while the mean training time per epoch measured in seconds for each network across all tasks is reported as an indication of efficiency.

2) *Results:* The results in Fig. 2 show that hPGA-DP, regardless of using U-Net or Transformer for denoising, consistently outperforms baseline policies that rely solely on these backbones. hPGA-DP achieves strong performance within 100 epochs on most tasks, with the exception of Mug, which involves multi-step interactions and requires more training. Notably, hPGA-U, despite having fewer parameters, often surpasses hPGA-T in performance.

hPGA-DP also demonstrates greater training efficiency. Despite that hPGA-DP takes more time to train per epoch according to the MET column of the bottom left table on Fig. 2, it requires many fewer training epochs to converge. For example, in the Stack task (bottom-right plot, Fig. 2), hPGA-DP variants reach high success rates within about 30 epochs, while U-Net-only and Transformer-only baselines require roughly three times more training epochs to match this level.

Policies using P-GATr as the denoising network fail across all tasks due to extremely slow convergence. As corroborated by the concurrent work [48], training P-GATr for effective



Network	Lift	Can	Stack	Square	Mug	MET
U-Net (State)	0.54	0.48	0.22	0.50	0.48	3.12
Trans. (State)	0.18	0.42	0.62	0.56	0.32	3.46
P-GATr (State)	0.00	0.00	0.00	0.00	0.00	23.49
U-Net (6D)	0.48	0.42	0.20	0.46	0.42	<b>3.08</b>
Trans. (6D)	0.12	0.44	0.60	0.54	0.30	3.49
P-GATr (6D)	0.00	0.00	0.00	0.00	0.00	23.75
hPGA-U (State)	0.98	0.96	<b>0.98</b>	<b>0.84</b>	<b>0.72</b>	4.35
hPGA-T (State)	<b>1.00</b>	<b>0.98</b>	0.82	0.70	0.62	4.61
hPGA-U (6D)	0.96	0.90	0.96	0.78	0.68	4.42
hPGA-T (6D)	0.98	0.92	0.84	0.68	0.60	4.66

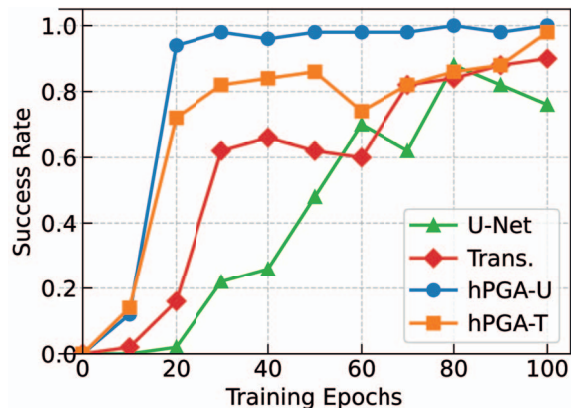


Fig. 2: **Top**: simulation tasks in robosuite, with colored 3D bounding boxes indicating task-relevant objects. **Bottom left**: success rates for diffusion policies with different network backbones for various tasks, and mean epoch training time (MET) for each network on all tasks together. **Bottom right**: plot of success rate for state-based policies with U-Net, Transformer, hPGA-U, and hPGA-T for 100 training epochs of the Stack task.

denoising requires at least seven days on high-end GPUs, rendering it far less practical than both hPGA-DP and standard diffusion backbones.

3) *Ablation Studies*: To further analyze the design of hPGA-DP, we conduct two ablation studies: (1) to identify an effective range for the action decoder loss masking threshold  $\eta$ , and (2) to evaluate whether the performance gains stem primarily from the encoder-denoiser-decoder layout. All experiments are conducted on the Stack task, which converges most rapidly among the benchmarks. Each configuration is evaluated over 5 trials, with 50 rollouts per trial. We report mean and standard deviation of the success rate.

For the  $\eta$  ablation, we test 10 values from 0.0 to 1.0 in increments of 0.1 for both hPGA-U and hPGA-T, using ground-truth object states. We additionally include two intermediate values in regions showing steep performance drops, resulting in 12 variants per model (top plot, Fig. 3). Results show that both hPGA-U and hPGA-T are robust to the choice of  $\eta$ : hPGA-U performs well for  $\eta \in [0.05, 0.75]$ , while hPGA-T maintains stable performance within  $\eta \in [0.05, 0.95]$ . The low variance across trials reflects consistency in training.

This robustness aligns with observations by Ho et al. [13], which suggest that most noise is removed in early denoising steps. Consequently, action latents quickly acquire a coarse geometric structure sufficient for effective decoder training, even if not yet executable. High values of  $\eta$  still provide meaningful gradients, while very low  $\eta$  limits decoder up-

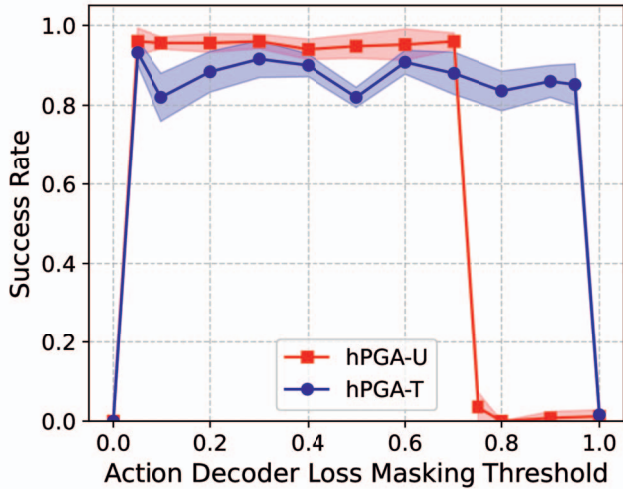
dates since most samples are excluded.

To evaluate the impact of network layout, we perform a second ablation by varying the encoder and decoder while keeping the denoiser fixed as either Transformer or U-Net. We compare three encoder-decoder types: MLP, Transformer, and P-GATr. The P-GATr variants correspond to hPGA-T and hPGA-U. MLP and Transformer are included due to their widespread use in prior works as encoder and decoder [44, 19, 9], while U-Net is typically only used as backbone [31, 32, 23]. For MLP and Transformer variants, we use standard MSE loss [6], as they do not operate on multivectors and thus do not require loss masking or  $\eta$ .

As shown in Fig.3, using MLP or Transformer as the encoder and decoder does not yield significant improvements over their baselines in Fig.2. This suggests that the gains from hPGA-DP arise not simply from its layout, but from the integration of P-GATr and its geometry-aware training strategy.

## B. Real-World Experiments

1) *Experimental Settings*: To further evaluate hPGA-DP, we conduct real-world experiments using a dual-arm setup (top-left image, Fig. 4) consisting of two 7-DOF xArm7 robots mounted on 1-DOF linear actuators. One arm is equipped with an Intel RealSense D435i depth camera, while the other is fitted with a parallel gripper. The system operates in a look-at end-effector space, where the camera's orientation is excluded from the state-action space by dy-



	<i>MLP</i>	<i>Trans.</i>	<i>P-GATr</i>
<u>U-Net</u>	0.24±0.06	0.26±0.04	<b>0.96±0.02</b>
<u>Trans.</u>	0.58±0.04	0.62±0.04	<b>0.88±0.05</b>

Fig. 3: **Top**: Success rate of hPGA-DP under different decoder loss masking thresholds  $\eta$ , where solid line denotes the mean and shaded region indicates the standard deviation. **Bottom**: Performance of diffusion policies with various combinations of backbone (underlined) and *encoder & decoder* (italicized).

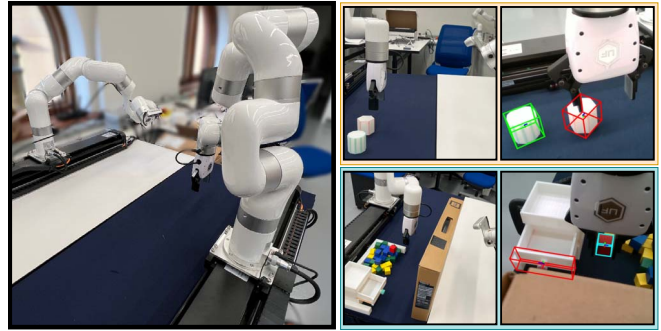
namically adjusting it through inverse kinematics constraints to maintain focus on the gripper [35, 26].

We evaluate our approach on two real-world tasks, visualized in Fig. 4. The block stacking task (Block Stack.) requires placing a non-cuboid block onto another, while the drawer interaction task (Drawer Inter.) involves clearing a visual occlusion, retrieving a red cube, inserting it into a drawer, and closing the drawer. The blocks and drawer are 3D printed using meshes from Lee et al. [16] and Heo et al. [12], which are also used as ground-truth meshes for evaluation. Each task is trained on a dataset of 200 demonstration rollouts.

The evaluated conditions are the same as the simulation experiment with the exception of those that received ground-truth state information as that is not available in the real-world. Instead, all conditions inferred object poses using the same 6D pose estimation strategy employed in the simulation experiments—namely, PRISM-DP [36] with a FoundationPose backend [43]. An example of pose tracking results from a single frame is shown in the top-right panel of Fig. 4.

Training and inference are performed on the same workstation used for simulation. We report success rate and cumulative training time as performance metrics. Note that training time includes only the forward and backward passes during epoch training, excluding data loading and sampling overhead.

2) *Results*: As shown in the table of Fig. 4, hPGA-DP achieves significantly higher success rates compared to other network backbones trained for the same number of epochs. While hPGA-DP requires more training time than



Network	Block Stack.		Drawer Inter.	
	SR	CT	SR	CT
U-Net	0.43	<b>9.85</b>	0.27	<b>14.28</b>
Trans.	0.37	12.03	0.40	15.87
P-GATr	0.00	92.42	0.00	119.63
hPGA-U	<b>0.97</b>	16.25	<b>0.90</b>	22.53
hPGA-T	0.93	17.69	0.87	26.81

Fig. 4: **Top left**: the dual-arm system for real-world experiments. **Top right**: top and bottom row show the block stacking task and drawer interaction task respectively. **Bottom**: results for real-world experiments. SR: success rate, CT: cumulative training time measured in minutes.

Transformer and U-Net baselines, those baselines need to be trained for twice as many epochs to match hPGA-DP’s performance, which results in 21% to 36% higher total training time. These findings are consistent with simulation results, demonstrating that hPGA-DP offers superior performance and training efficiency over traditional diffusion backbones.

## VI. DISCUSSION

In this work, we introduced hPGA-DP, a hybrid diffusion policy that successfully integrates PGA to embed strong geometric inductive biases into robot learning. Our architecture uses P-GATr for state encoding and action decoding alongside a conventional denoising module. This design, combined with a staged supervision strategy for the decoder, significantly improves task performance and convergence speed over standard baselines by harnessing PGA’s geometric reasoning while maintaining practical training efficiency.

While promising, our approach has limitations that suggest avenues for future work. Although hPGA-DP converges in fewer epochs than traditional diffusion backbones, each epoch takes slightly longer to train. This inefficiency likely stems from our current PyTorch-based implementation, where the default backward pass may poorly handle the intricate blade-wise interactions in PGA. This could be addressed by developing custom compute kernels using lower-level frameworks such as Triton [38] to accelerate the PGA operations. Solving this challenge would further broaden the applicability of using geometric algebras in robot learning.

## REFERENCES

- [1] Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*.

- [2] Cem Bilaloglu, Tobias Löw, and Sylvain Calinon. Tactile ergodic control using diffusion and geometric algebra. *arXiv preprint arXiv:2402.04862*, 2024.
- [3] Johann Brehmer, Pim De Haan, Sönke Behrends, and Taco S Cohen. Geometric algebra transformer. *Advances in Neural Information Processing Systems*, 36: 35472–35496, 2023.
- [4] Johann Brehmer, Víctor Bresó, Pim de Haan, Tilman Plehn, Huilin Qu, Jonas Spinner, and Jesse Thaler. A lorentz-equivariant transformer for all of the lhc. *arXiv preprint arXiv:2411.00446*, 2024.
- [5] Oscar Carbajal-Espinosa, Leobardo Campos-Macías, and Miriam Díaz-Rodríguez. Fika: a conformal geometric algebra approach to a fast inverse kinematics algorithm for an anthropomorphic robotic arm. *Machines*, 12(1):78, 2024.
- [6] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.
- [7] Leo Dorst and Steven De Keninck. A guided tour to the plane-based geometric algebra pga. URL <https://bivector.net/PGA4CS.html>, 2022.
- [8] Leo Dorst, Daniel Fontijne, and Stephen Mann. *Geometric algebra for computer science (revised edition): An object-oriented approach to geometry*. Morgan Kaufmann, 2009.
- [9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [10] Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023.
- [11] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
- [12] Minh Heo, Youngwoon Lee, Doohyun Lee, and Joseph J Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. *The International Journal of Robotics Research*, page 02783649241304789, 2023.
- [13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [14] Zhi Hou, Tianyi Zhang, Yuwen Xiong, Hengjun Pu, Chengyang Zhao, Ronglei Tong, Yu Qiao, Jifeng Dai, and Yuntao Chen. Diffusion transformer policy. *arXiv preprint arXiv:2410.15959*, 2024.
- [15] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.
- [16] Alex X Lee, Coline Manon Devin, Yuxiang Zhou, Thomas Lampe, Konstantinos Bousmalis, Jost Tobias Springenberg, Arunkumar Byravan, Abbas Abdolmaleki, Nimrod Gileadi, David Khosid, et al. Beyond pick-and-place: Tackling robotic stacking of diverse shapes. In *5th Annual Conference on Robot Learning*, 2021.
- [17] Kin Long Kelvin Lee, Carmelo Gonzales, Matthew Spellings, Mikhail Galkin, Santiago Miret, and Nalini Kumar. Towards foundation models for materials science: The open matsci ml toolkit. In *Proceedings of the SC'23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, pages 51–59, 2023.
- [18] Qifan Liu and Wenming Cao. Geometric algebra graph neural network for cross-domain few-shot classification. *Applied Intelligence*, 52(11):12422–12435, 2022.
- [19] Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59):eabg5810, 2021.
- [20] Tobias Löw and Sylvain Calinon. Geometric algebra for optimal control with applications in manipulation tasks. *IEEE Transactions on Robotics*, 39(5):3586–3600, 2023.
- [21] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.
- [22] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.
- [23] Ilias Mitsouras, Eleftherios Tsonis, Paraskevi Tzouveli, and Athanasios Voulodimos. U-sketch: An efficient approach for sketch to image diffusion models. *arXiv preprint arXiv:2403.18425*, 2024.
- [24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
- [25] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.
- [26] Daniel Rakita, Bilge Mutlu, and Michael Gleicher. 2022.

- An autonomous dynamic camera method for effective remote teleoperation. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 325–333, 2018.
- [27] Moritz Reuss, Jyothish Pari, Pulkit Agrawal, and Rudolf Lioutikov. Efficient diffusion transformer policies with mixture of expert denoisers for multitask learning. *arXiv preprint arXiv:2412.12953*, 2024.
- [28] Martin Roelfs and Steven De Keninck. Graded symmetry groups: plane and simple. *Advances in Applied Clifford Algebras*, 33(3):30, 2023.
- [29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5–9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.
- [30] David Ruhe, Johannes Brandstetter, and Patrick Forré. Clifford group equivariant neural networks. *Advances in Neural Information Processing Systems*, 36:62922–62990, 2023.
- [31] Neha Sharma, Sheifali Gupta, Deepika Koundal, Sultan Alyami, Hani Alshahrani, Yousef Asiri, and Asadullah Shaikh. U-net model with transfer learning model as a backbone for segmentation of gastrointestinal tract. *Bioengineering*, 10(1):119, 2023.
- [32] Chenyang Si, Ziqi Huang, Yuming Jiang, and Ziwei Liu. Freeu: Free lunch in diffusion u-net. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4733–4743, 2024.
- [33] Guangzhen Sun and Ye Ding. An analytical method for sensitivity analysis of rigid multibody system dynamics using projective geometric algebra. *Journal of Computational and Nonlinear Dynamics*, 18(11), 2023.
- [34] Guangzhen Sun and Ye Ding. High-order inverse dynamics of serial robots based on projective geometric algebra. *Multibody System Dynamics*, 59(3):337–362, 2023.
- [35] Xiatao Sun, Francis Fan, Yinxing Chen, and Daniel Rakita. A comparative study on state-action spaces for learning viewpoint selection and manipulation with diffusion policy. *arXiv preprint arXiv:2409.14615*, 2024.
- [36] Xiatao Sun, Yinxing Chen, and Daniel Rakita. Prism-dp: Spatial pose-based observations for diffusion-policies via segmentation, mesh generation, and pose tracking. *arXiv preprint arXiv:2504.20359*, 2025.
- [37] Xiatao Sun, Shuo Yang, Yinxing Chen, Francis Fan, Yiyang Liang, and Daniel Rakita. Dynamic rank adjustment in diffusion policies for efficient and flexible training. *arXiv preprint arXiv:2502.03822*, 2025.
- [38] Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19, 2019.
- [39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [40] Dexin Wang, Chunsheng Liu, Faliang Chang, and Yichen Xu. Hierarchical diffusion policy: manipulation trajectory generation via contact guidance. *IEEE Transactions on Robotics*, 2025.
- [41] Dian Wang, Stephen Hart, David Surovik, Tarik Kelestemur, Haojie Huang, Haibo Zhao, Mark Yeatman, Jiuguang Wang, Robin Walters, and Robert Platt. Equivariant diffusion policy. *arXiv preprint arXiv:2407.01812*, 2024.
- [42] Qianhao Wang, Yinqian Sun, Enmeng Lu, Qian Zhang, and Yi Zeng. Mtdp: Modulated transformer diffusion policy model. *arXiv preprint arXiv:2502.09029*, 2025.
- [43] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17868–17879, 2024.
- [44] Yuwei Wu, Xiatao Sun, Igor Spasojevic, and Vijay Kumar. Deep learning for optimization of trajectories for quadrotors. *IEEE Robotics and Automation Letters*, 9(3):2479–2486, 2024.
- [45] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tiejian Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pages 10524–10533. PMLR, 2020.
- [46] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.
- [47] Jianqi Zhong and Wenming Cao. Geometric algebra-based multiview interaction networks for 3d human motion prediction. *Pattern Recognition*, 138:109427, 2023.
- [48] Tao Zhong and Christine Allen-Blanchette. Gagrasp: Geometric algebra diffusion for dexterous grasping. *arXiv preprint arXiv:2503.04123*, 2025.
- [49] Dongyang Zhu, Zhonghai Zhang, and Duanling Li. A conformal geometric algebra method for inverse kinematics analysis of 6r robotic arm. *Journal of Mechanisms and Robotics*, pages 1–34, 2025.
- [50] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.