NSactor: Next-Scale Feature Refining for Multi-Task Manipulation Learning(under drafting)

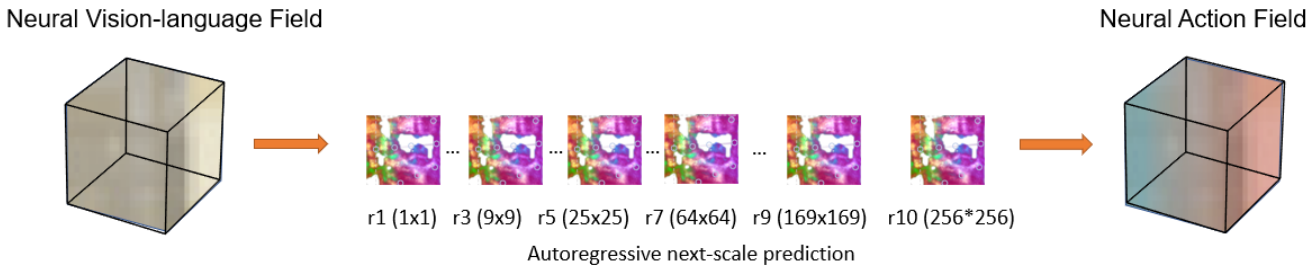

Fig. 1. We propose NSactor, a scale-autoregressive action predictor that utilizes fused vision-language features. Unlike standard vision transformers, which generate tokens in raster-scan order, NSactor employs a next-scale prediction approach, generating vision-language features for the subsequent scale based on the entire feature map of the current scale. This scale-autoregressive modeling framework not only preserves the spatial structure—critical for precision in fine manipulation tasks—but also recursively refines it for enhanced performance.

***Abstract*— Generalizable robotic manipulation has long posed a significant challenge due to the demanding requirements for precision, task diversity, and robustness against environmental noise. This paper presents NSactor, a multi-task policy learning framework that harnesses the synergistic capabilities of expert demonstrations and language instructions for effective manipulation control. The framework refines action predictions autoregressively through a next-scale prediction sequence, maintaining spatial structural knowledge using a transformer-based architecture. Inspired by principles of visual autoregressive modeling (VAR), NSactor incrementally reconstructs vision-language features at progressively larger scales, recursively enhancing the reliability and precision of action predictions. Environment distillation is achieved by optimizing neural feature fields under the supervision of a vision-language foundation model. Leveraging the inherent generalizability of foundation models, NSactor requires only minimal expert demonstrations to develop a robust understanding of the environment. Experimental results validate the efficacy of NSactor in learning a unified manipulation policy across ten distinct tasks in simulation, achieving a xx% improvement in success rate and generalizability over prior NeRF-based multi-task learners. Moreover, in real-world robotic applications, NSactor demonstrated consistent robustness and reliability across three diverse tasks. Project website: in process**

I. INTRODUCTION

The ability to perform multi-task robotic manipulation in complex environments is becoming increasingly essential for generalist robots, particularly household robots that must execute a wide range of robust and versatile tasks. Recent advancements in multi-task learning [1] within the imitation learning paradigm have demonstrated the feasibility of training a single model capable of performing diverse tasks. Unlike reinforcement learning, imitation learning eliminates the need for densely labeled rewards or online/on-policy interactions, making it a more efficient approach for robotic task learning. Nevertheless, significant challenges remain in realizing the vision of a general-purpose home robot. (i) Developing a generalizable multi-task policy from scratch necessitates large-scale datasets to capture comprehensive spatial features, which is a substantial hurdle in real-world scenarios where obtaining expert demonstrations is both time-intensive and laborious. (ii) Household robots must achieve a high degree of precision to handle delicate tasks, such as manipulating small or fragile items, reliably. Addressing these challenges requires an integrated learning framework capable of efficiently distilling multimodal environmental information, maximizing the utility of limited expert demonstrations, and/or effectively leveraging spatial and language representations from pretrained foundation models. Such a framework must incorporate structured and efficient vision-language features, enabling a predictive model to accurately capture subtle spatial discrepancies with limited expert demonstrations and retain critical environmental and instruction features during the action generation.

Addressing the challenge of efficiently distilling environmental knowledge with limited data has seen significant progress, driven by the evolution of spatial feature representations from 2D vision features to point clouds and, more recently, to NeRF-based 3D spatial features. Early approaches, such as Gato [2] and BC-Z [3], directly map 2D images to 6-DoF actions. However, these methods require extensive data collection over weeks or even months

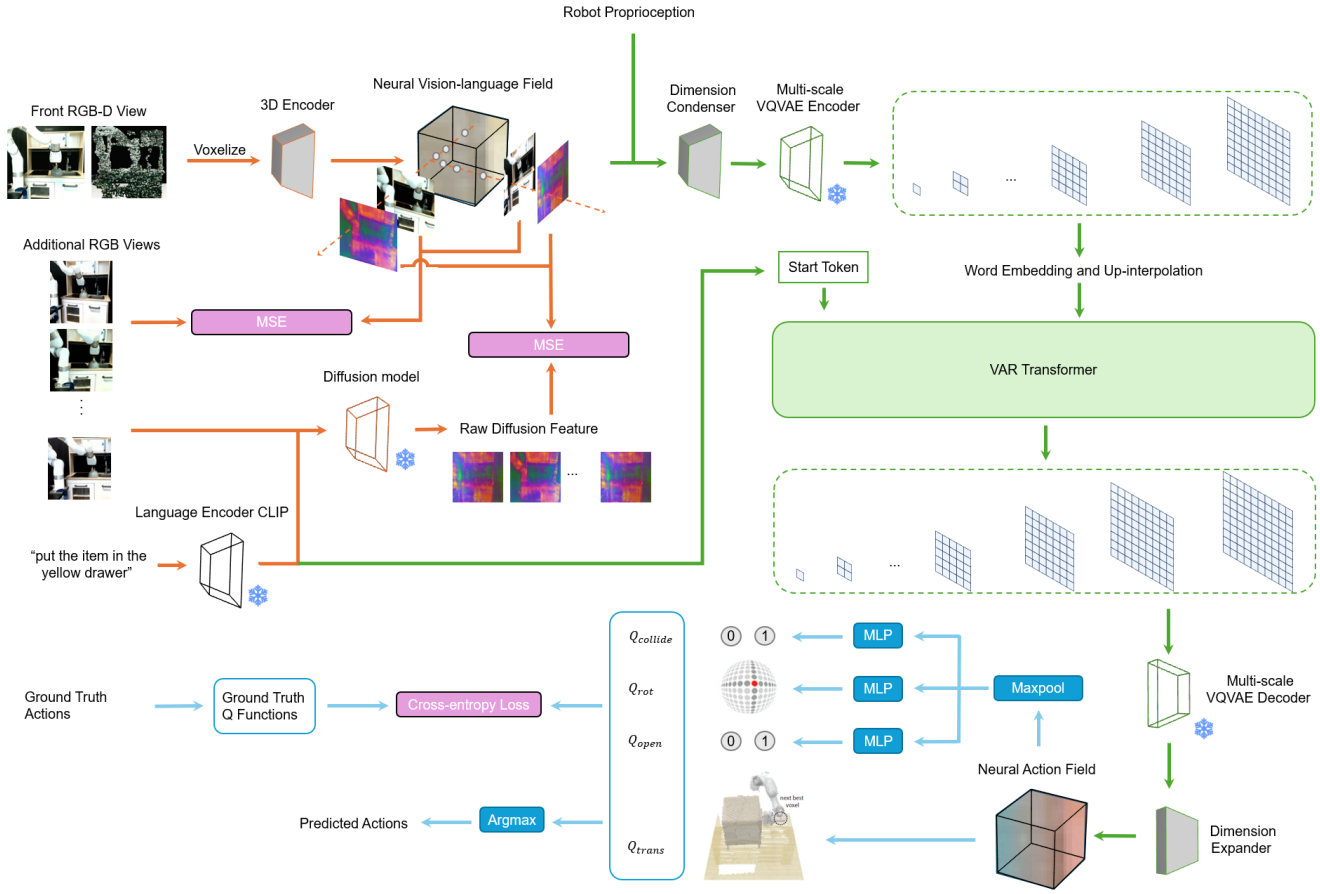

Fig. 2. The diagram illustrates the overall architecture of the proposed NSactor framework. The raw inputs consist of an RGB-D image, robot proprioception data, and a language instruction, which are processed through genralizable neural feature field and action prediction modules. The orange segment represents the NeRF-based representation optimization, the green segment denotes the autoregressive refinement at the subsequent scale, and the blue flow highlights the extraction of 3D actions from the volumetric latent space. To optimize the neural fields, multiple RGB views are required, while ground truth actions are utilized to compute the behavior cloning loss.

to train effective multi-task policies. CoTPC [4] proposed a different approach by extracting subskill demonstrations from 2D image sequences and leveraging them as guidance for action prediction, maximizing the utility of image data for policy generation. Building on these advancements, voxel-based representations have emerged as a powerful alternative. For instance, C2FARM [5] recursively constructs voxelized observations from 2D images and point clouds, refining resolution from coarse to fine. PERACT [1], on the other hand, employs triangulation of multiple RGBD images from different viewpoints to create voxel-based formulations. This approach offers numerous advantages, including natural fusion of multi-view observations, robust perceptual representations of actions, and enhanced data augmentation in 6-DoF. These capabilities enable the learning of generalizable skills by focusing on diverse, broad-spectrum multi-task data rather than narrow datasets. Recently, neural radiance fields (NeRF) [6] have become a preferred choice for spatial feature understanding. NeRFs combine the strengths of voxelized representations with precise 3D spatial knowledge, offering significant benefits for robotic manipulation tasks that require robust 3D latent features for real-world interactions. Initially adopted in reinforcement learning [7], NeRFs have been used to supervise state representation learning, yielding improved performance. Extensions such as GNFactor [8] and DNAct [9] further integrate object semantics and language instructions through optimization, resulting in generalizable neural feature fields suitable for multi-task manipulation learning.

The quality of robotic manipulation heavily relies on the ability of the action predictor to transform latent features into effective actions. Transformers [10] have emerged as a popular backbone architecture for action prediction due to their proven excellence in natural language processing and computer vision tasks. In an autoregressive framework, Behavior Transformers [11] discretize continuous actions into bins and leverage the context-aware, multi-token prediction capabilities of transformer-based sequence models to predict multimodal continuous actions. Similarly, CoTPC [4] employs a transformer decoder with learnable prompt tokens and a hybrid masking strategy to couple continuous action and subskill predictions. However, when handling 3D latent features, a more efficient transformer is required to manage the increased data dimensionality. To address this, PERACT [1] uses a PerceiverIO Transformer [12], encoding

language instructions and RGB-D voxel observations into a small set of latent vectors, enabling the processing of high-dimensional inputs (up to 1 million voxels). Building on this approach, GNFactor [8] successfully trained a multi-task policy that maps 3D neural feature fields to 3D action spaces. Beyond transformers, diffusion models [13] have gained attention for action prediction, leveraging their superior performance in image synthesis, natural language processing, and multi-modal learning of robotic demonstrations. Works such as Diffusion Policy [14, 15] employ diffusion models to learn expressive generative policies capable of capturing multiple action modes, thereby enhancing success rates. However, these studies primarily focus on relatively simple single-task scenarios, and the extensive inference time of diffusion models presents a significant limitation for real-world applications requiring both accuracy and low-latency predictions. To enhance efficiency, the state-of-the-art algorithm DNAct [9] bypasses direct use of diffusion models for action inference. Instead, it employs them to derive multi-modal representations and jointly optimize a policy network, ensuring more stable training while avoiding the computational overhead associated with diffusion model inference.

Despite the advances in demonstration learning and action prediction, many existing algorithms remain limited to single-task scenarios, and their success rates in multi-task settings leave significant room for improvement. To address this gap, we propose a more robust action predictor inspired by the concept of next-scale prediction from VAR [15]. As a transformer-based image generative model, VAR was the first to demonstrate that GPT-style autoregressive models could comprehensively outperform diffusion transformers [16, 17] across multiple dimensions of image generation. We explore its potential in enhancing multi-task robotic learning, leveraging its ability to preserve spatial structure in high-dimentional data during autoregressive refinement. Unlike standard autoregressive modeling, which generates sequential visual tokens in a raster-scan order [18] and thus sacrifices bidirectional feature correlations, next-scale prediction generates tokens progressively from lower to higher resolutions while retaining the full context of the scene. Therefore, the standard autoregressive approach, as employed by PERACT [1], may inadvertently compromise structural priors in action predictions, akin to the limitations of VQGAN [19] in image generation.

To retain high-dimentiaonal correlations in spatial features, we introduce NSactor, a novel framework designed to enhance action distillation through structural autoregressive modeling of NeRF-based volume latents. Our approach employs advanced neural field representations of spatial structure and language instructions, drawing from DNAct [9]. To further enhance the generalizability of latent features, we employ language and vision foundation models, specifically Stable Diffusion [20] and CLIP [21], for vision-language token generation. Finally, the action predictor utilizes a GPT-like backbone with a scale-autoregressive design to process the generalizable neural feature field and output precise 3D actions. This design enables NSactor to address the limitations of existing methods and achieve superior performance in multi-task robotic learning. As far as we know, this is the first work investigating the benefits of scale-autoregression of high dimention features in robot learing. Figure 2 summarizes the entire framework in DSactor. It illustrates how the latent space feature and scale autoregression is constructed and trained. During inference, the neural feature field will not be optimized, instead, it will be used as a forward encoder. Through extensive experiments in both simulation and real robot setting, NSactor is able to learn policies achieving higher success rate not only in the training environments but also in the environments where novel objects and arrangement is present. Notablely, NSactor achieves a xxx improvement in simulation and a xxx improvement in real-world robot experiments. In scenarios with novel objects, our ablation study shows NSactor outperforming baselines by xxx.

II. METHOD

In this section, we elaborate on our proposed method, NSactor. The NSactor learns a language-conditioned multi-task policy by leveraging 3D encoder from neural featural rendering and taking the next-scale prediction paradigm in action generation.

A. Demonstrations and Keyframes

We consider a dataset $\mathcal{D} = \{\zeta_1, \zeta_2, \ldots, \zeta_n\}$, comprising n expert demonstrations, where each demonstration is associated with a corresponding set of English language instructions $\mathcal{G} = \{l_1, l_2, \ldots, l_n\}$. These demonstrations are generated using a motion-planner to guide the expert through intermediate target poses. Each individual demonstration ζ consists of a sequence of continuous actions $\mathcal{A} = \{a_1, a_2, \ldots, a_t\}$ and their corresponding observations $\mathcal{O} = \{\tilde{o}_1, \tilde{o}_2, \ldots, \tilde{o}_t\}$. Each action a includes the 6-DoF pose, the state of the gripper (open or closed), and a flag indicating whether collision avoidance was employed by the motion-planner: $a = \{a_{\text{trans}}, a_{\text{open}}, a_{\text{collide}}\}$. The observations $\tilde{o}$ are composed of a front RGB-D image and several RGB images from different viewpoints. For simulated environments, we utilize 19 camera views to generate RGB images, while for real-world experiments, only xx RGB images are used to collect data.

To address the challenges of computational complexity and data inefficiency in continuous action prediction, we redefine the task of learning from demonstrations in robotic manipulation as a keyframe-prediction problem [1, 5]. Unlike continuous prediction, this reformulation focuses on identifying keyframes from expert demonstrations using specific criteria: a frame qualifies as a keyframe when joint velocities approach zero, and the gripper consistently remains in an open state. By leveraging these keyframes, the model learns to predict the subsequent keyframe based on current multiview observations, effectively simplifying the overall learning process. This formulation delegates the internal procedures to the RRT-connect motion planner [22]

in simulation and Linear motion planner in real-world xx robot.

The transformation from continuous to keyframe-based prediction retains similarities to conventional learning-from-demonstrations workflows but introduces a discrete representation of robot actions. Actions of the robot arm equipped with a gripper are represented through four components: the translation $a_{\text{trans}} \in R^3$, the rotation $a_{\text{rot}} \in R^{(360/5)\times 3}$, the openness of the gripper $a_{\text{open}} \in [0, 1]$, and the collision avoidance indicator $a_{\text{collision}} \in [0, 1]$. The rotation is discretized into 72 bins, each covering a 5-degree interval per rotational axis.

B. NSactor Agent

NSactor takes in keyframes of demonstrations and language instruction, and then outputs a discretized translation, rotation, and gripper open action for the next keyframe timestep. Its architecture consists of three key blocks: 3D demonstration and language instruction embedding, 3D action space learning, and action extraction, while the second part is our core contribution.

Vision-language feature space is built through the optimization of generaliable neural feature fields [8]. Specifically, front RGB-D view is vocelized and encoded into a voxel grid $v \in R^{100^3\times 128}$. Assume that $v_x \in R^{128}$ denotes a sampled 3D feature in the voxel grid v with trilinear interpolation, the generaliable neural feature fields are optimized to encode three functions: (i) a density function $\sigma(\mathbf{x}, \mathbf{v_x}) : R^{3+128} \to R_+$, which maps a 3D point $\mathbf{x}$ and its corresponding feature vector $\mathbf{v_x}$ to a density value, (ii) an RGB function $c(\mathbf{x}, \mathbf{d}, \mathbf{v_x}) : R^{3+3+128} \to R^3$, which determines the color of a 3D point based on its position $\mathbf{x}$, viewing direction $\mathbf{d}$, and feature vector $\mathbf{v_x}$, and (iii) a vision-language embedding function $f(\mathbf{x}, \mathbf{d}, \mathbf{v_x}) : R^{3+3+128} \to R^{512}$, which maps the point and feature vector to an embedding space.

Given a camera ray $\mathbf{r}(t) = \mathbf{o}+t\mathbf{d}$, originating from $\mathbf{o} \in R^3$ and extending along direction $\mathbf{d}$ within depth bounds $[t_n, t_f]$, we compute the estimated color and embedding as:

$$\hat{\mathbf{C}}(\mathbf{r}, \mathbf{v}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t), \mathbf{v_x}(t))c(\mathbf{r}(t), \mathbf{d}, \mathbf{v_x}(t))dt, \quad (1)$$

$$\hat{\mathbf{F}}(\mathbf{r}, \mathbf{v}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t), \mathbf{v_x}(t))f(\mathbf{r}(t), \mathbf{d}, \mathbf{v_x}(t))dt, \quad (2)$$

where $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(s)ds\right)$ is the accumulated transmittance along the ray. Numerical quadrature approximates the integrals in our implementation. The framework optimizes reconstruction of RGB images and vision-language embeddings from Stable Diffusion across diverse scenes by minimizing the loss:

$$\mathcal{L}_{\text{recon}} = \sum_{\mathbf{r}\in\mathcal{R}} \|\mathbf{C}(\mathbf{r}) - \hat{\mathbf{C}}(\mathbf{r})\|_2^2 + \lambda_{\text{feat}}\|\mathbf{F}(\mathbf{r}) - \hat{\mathbf{F}}(\mathbf{r})\|_2^2, \quad (3)$$

where $\mathbf{C}(\mathbf{r})$ and $\mathbf{F}(\mathbf{r})$ represent ground truth color and vision-language embeddings, respectively. The loss incorporates a weighting factor λ_{feat} to balance embedding reconstruction. To improve efficiency, we sample rays from target views and adopt a coarse-to-fine hierarchical structure inspired by NeRF, augmented with depth-guided sampling to enhance results in the finer network [23]. Notably, the neural feature fields remain fixed during inference and are not re-optimized. As the volumetric representation v is specifically trained to facilitate action generation, its optimization is conducted concurrently with the training of the action prediction model.

The action predictor is a GPT-like transformer employing a scale-autoregressive modeling paradigm. To enable efficient autoregression across scales, the voxel grid is first condensed into a size of $R^{256^2\times 3}$, denoted as v_2. This condensed representation, v_2, is then tokenized using a multi-scale quantization autoencoder, encoding it into K multi-scale discrete token maps $R = (r_1, r_2, \ldots, r_K)$, each with progressively higher resolutions $h_k \times w_k$. The process culminates in r_K, which matches the resolution of the condensed feature map at 256×256.

The multi-scale quantization autoencoder adopts the same architecture as VQVAE [19] but with a modified multi-scale quantization layer. During training, the multi-scale VQVAE encoder is frozen, utilizing pre-trained parameter values provided by VAR [15]. The resulting multi-scale discrete token maps are then passed to the VAR transformer, a decoder-only transformer modeled after GPT-2, enhanced with adaptive normalization (AdaLN). The autoregressive likelihood via VAR transformer is formulated as follows:

$$p(r_1, r_2, \ldots, r_K) = \prod_{k=1}^{K} p(r_k \mid r_1, r_2, \ldots, r_{k-1}), \quad (4)$$

where each unit $r_k \in [V]^{h_k\times w_k}$ represents the token map at scale k with $h_k \times w_k$ tokens, and the sequence $(r_1, r_2, \ldots, r_{k-1})$ provides the "prefix" for r_k. At the k-th step, the distributions over the $h_k \times w_k$ tokens in r_k are generated simultaneously. During training, a block-wise causal attention mask ensures r_k only focuses on its prefix $r_{\leq k}$. For inference, kv-caching removes the need for masking, enabling efficient computation.

The details of the VQVAE encoder and VAR transformer can be found in [15] and our source code. In our implementation, the class condition label in VAR is replaced with a language instruction token generated by CLIP [21], which serves as both the start token and the condition for AdaLN. For the resolutions at each scale, we use patch numbers of 1, 2, 3, 4, 5, 6, 8, 10, 13, and 16 as the baseline for the autoregressive sequence scale, with each patch having a resolution of 16×16. A more detailed investigation of the autoregressive scales is provided in the ablation study section.

The discrete token map at the final scale is utilized to construct the latent 3D action space. First, a multi-scale VQVAE decoder retrieves features from the discrete token map, using the same codebook as the multi-scale VQVAE

encoder. These features are then restored to their 3D dimensions through a dimension expander, which employs the inverse structure of the dimension condenser. The resulting voxel grid, v_{ns}, is added to v via a skip connection to form the neural action field. Unlike NeRF, action rendering is achieved by reshaping and max-pooling [5]. For translation, these features are reshaped into the original 3D voxel grid (100^3) to compute a Q-function representing action values. For rotation, gripper open state, and collision, the features are aggregated using max-pooling and processed through linear layers to produce their corresponding Q-functions. The optimal action $\mathcal{T}$ is obtained by maximizing each Q-function as follows:

$$\begin{aligned}
\mathcal{T}_{\text{trans}} &= \arg\max_{(x,y,z)} Q_{\text{trans}}((x,y,z) \mid \mathbf{v}, \mathbf{1}), \\
\mathcal{T}_{\text{rot}} &= \arg\max_{(\psi,\theta,\phi)} Q_{\text{rot}}((\psi,\theta,\phi) \mid \mathbf{v}, \mathbf{1}), \\
\mathcal{T}_{\text{open}} &= \arg\max_{\omega} Q_{\text{open}}(\omega \mid \mathbf{v}, \mathbf{1}), \\
\mathcal{T}_{\text{collide}} &= \arg\max_{\kappa} Q_{\text{collide}}(\kappa \mid \mathbf{v}, \mathbf{1}),
\end{aligned} \tag{5}$$

where (x, y, z) represents the voxel position in the grid, (ψ, θ, ϕ) are discrete Euler angle rotations, ω denotes the gripper's open state, and κ is the collision indicator. The cross-entropy loss between predicted Q-functions and groundtruth Q-functions are used to optimize the behavior cloning performance:

$$\begin{aligned}
\mathcal{L}_{\text{action}} = &- E_{Y_{\text{trans}}}\left[\log \mathcal{V}_{\text{trans}}\right] - E_{Y_{\text{rot}}}\left[\log \mathcal{V}_{\text{rot}}\right] \\
&- E_{Y_{\text{open}}}\left[\log \mathcal{V}_{\text{open}}\right] - E_{Y_{\text{collide}}}\left[\log \mathcal{V}_{\text{collide}}\right],
\end{aligned} \tag{6}$$

where $\mathcal{V}_i = \text{softmax}(Q_i)$ computes normalized probabilities for $Q_i \in \{Q_{\text{trans}}, Q_{\text{rot}}, Q_{\text{open}}, Q_{\text{collide}}\}$, and $Y_i \in \{Y_{\text{trans}}, Y_{\text{rot}}, Y_{\text{open}}, Y_{\text{collide}}\}$ denotes the one-hot encoded ground truth labels. The comprehensive learning objective for GNFactor is expressed as:

$$\mathcal{L}_{\text{GNFactor}} = \mathcal{L}_{\text{action}} + \lambda_{\text{recon}} \mathcal{L}_{\text{recon}}, \tag{7}$$

where λ_{recon} adjusts the relative contribution of the reconstruction loss, ensuring compatibility between different objectives.

C. Training Details

NSACTOR is trained using supervised learning on discrete-time input-action pairs derived from a demonstration dataset. Each pair consists of voxel observations, language instructions, and corresponding keyframe actions, represented as $\{(v_1, l_1, k_1), (v_2, l_2, k_2), \ldots\}$. During training, a tuple is randomly sampled, and the agent is guided to predict the keyframe action k based on the given voxel observation and language instruction (v, l). The entire model consists of 86M parameters when the VAR transformer has a depth of 8, and the LAMB optimizer [24] is employed in training.

III. EXPERIMENTS

A. Experiment Setup

Simulation. All simulated experiments are conducted using RLBench [25], a comprehensive benchmark and learning environment for robotic tasks. The setup includes a 7-DoF Franka Emika Panda robotic arm with a parallel gripper mounted on a table for executing various tasks. Visual data is captured from four RGB-D cameras positioned at the front, left shoulder, right shoulder, and wrist, each providing images at a resolution of 128×128. Each task involves variations in properties such as shape and color, accompanied by descriptive text summaries. We select 10 challenging tasks conditioned on language instructions, with 50 demonstrations per task used for training. To achieve success with limited training data, the agent must acquire generalizable manipulation skills rather than simply memorizing the training demonstrations.

Real Robot.

Expert Demonstrations. For each RLBench task, we gather 50 demonstrations using a motion planner, with task variations sampled uniformly. Additionally, we collect xx demonstrations for each real robot task, utilizing a xx controller.

B. Simulation Results

Multi-Task Performance. We use the same benchmark tasks utilized by DNAct, so that we can still compare our model's performance with DNAct despite an exactly identical DNAct model being unrebuildable for us due to the currently inaccessible source code.

Generalization Performance. As the source code for DNAct is unavailable, we were unable to reconstruct an identical model for NSactor. Furthermore, the generalization tasks used by DNAct require specific environment parameters, as they are not standardized tasks within RLBench. Consequently, we were unable to reproduce either the DNAct model or the corresponding generalization tasks. Therefore, we do not compare NSactor's generalization performance with that of DNAct. **Ablation**

C. Real-Robot Results

IV. CONCLUSIONS

A conclusion section

APPENDIX

Appendixes

ACKNOWLEDGMENT

The

REFERENCES

[1] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*, 2023, pp. 785–799.

[2] S. Reed *et al.*, "A generalist agent," *arXiv preprint arXiv:2205.06175*, 2022.

[3] E. Jang *et al.*, "BC-Z: Zero-shot task generalization with robotic imitation learning," in *Conference on Robot Learning*, 2022, pp. 991–1002.

TABLE I

MULTI-TASK PERFORMANCE ON RLBENCH. WE EVALUATE 25 EPISODES FOR EACH CHECKPOINT ON 10 TASKS ACROSS 3 SEEDS AND REPORT THE SUCCESS RATES (%) OF THE FINAL CHECKPOINTS. OUR METHOD OUTPERFORMS THE MOST COMPETITIVE BASELINE PERACT [1], GNFACTOR [8], AND DNACT [9] WITH AN AVERAGE IMPROVEMENT OF XXX AND XXX.

Methods	**turn tap**	**drag stick**	**open fridge**	**put in drawer**	**sweep to dustpan**	**meat off grill**	**phone on base**	**place wine**	**slide block**	**put in safe**	**Average**
PerAct	57.3±6.1	14.7±6.1	4.0±6.9	16.0±13.9	4.0±6.9	77.3±14.0	98.7±2.3	7.6±6.1	29.3±22.0	48.0±26.2	35.6
GNFactor	56.0±14.4	68.0±38.6	2.7±4.6	12.0±6.9	61.3±6.1	77.3±3.9	96.0±6.9	8.0±6.9	21.3±6.1	30.7±6.1	43.3
DNAct	73.3±4.6	97.3±4.6	22.7±16.2	54.7±22.0	24.0±17.4	92.0±6.9	82.7±12.2	65.3±9.2	58.7±20.5	59.6±8.3	59.6
NSactor (under upgrading)	59±0	75±0	27±0	48±0	15±0	69±0	50±0	62±0	55±0	71±0	

TABLE II

WE ALSO REPORT THE SUCCESS RATES (%) OF THE BEST CHECKPOINTS FOR EACH TASK ACROSS 3 SEEDS. OUR METHOD ACHIEVES XXX AND XXX IMPROVEMENT COMPARED WITH PERACT AND GNFACTOR.

Methods	**turn tap**	**drag stick**	**open fridge**	**put in drawer**	**sweep to dustpan**	**meat off grill**	**phone on base**	**place wine**	**slide block**	**put in safe**	**Average**
PerAct	66.7±4.6	38.7±6.1	12.0±6.9	13.0±6.1	4.0±6.9	85.3±8.3	100.0±0.0	13.3±2.3	38.7±6.1	62.7±11.5	45.2
GNFactor	73.3±6.1	92.0±7.6	7.6±6.1	16.0±6.9	6.2±6.9	86.7±2.3	100.0±0.0	8.0±6.9	42.0±4.0	45.3±6.1	57.6
DNAct	85.3±4.6	100.0±0.0	28.0±10.6	70.7±12.9	72.0±6.9	96.0±4.0	89.3±8.4	72.0±6.9	84.0±8.0	74.7±6.1	74.7
NSactor (under upgrading)	86±0	89±0	31±0	59±0	34±0	73±0	53±0	66±0	62±0	75±0	

TABLE III

GENERALIZATION TO UNSEEN TASKS ON RLBENCH. WE EVALUATE 20 EPISODES FOR EACH TASK WITH THE FINAL CHECKPOINT ACROSS 3 SEEDS. WE DENOTE "L" AS A LARGER OBJECT, "S" AS A SMALLER OBJECT, "N" AS A NEW POSITION, AND "D" AS ADDING A DISTRACTOR. OUR METHOD OUTPERFORMS PERACT WITH AN AVERAGE IMPROVEMENT OF XXX.

Method / Task	**drag (D)**	**slide (L)**	**slide (S)**	**turn (N)**	**Average**
PerAct	6.6±4.7	33.3±4.7	5.0±4.1	18.3±6.2	18.0
GNFactor	46.7±30.6	25.0±4.1	6.7±6.2	28.3±2.4	28.3
NSactor (under upgrading)	42±0	19±0	11±0	32±0	

[4] Z. Jia, V. Thumuluri, F. Liu, L. Chen, Z. Huang, and H. Su, "Chain-of-thought predictive control," *arXiv preprint arXiv:2304.00776*, 2023.

[5] S. James, K. Wada, T. Laidlow, and A. J. Davison, "Coarse-to-fine Q-attention: Efficient learning for visual robotic manipulation via discretisation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13739–13748.

[6] B. Mildenhall *et al.*, "NeRF: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no.

1, pp. 99–106, 2021.

[7] D. Driess, I. Schubert, P. Florence, Y. Li, and M. Toussaint, "Reinforcement learning with neural radiance fields," in *Advances in Neural Information Processing Systems*, vol. 35, pp. 16931–16945, 2022.

[8] Y. Ze *et al.*, "GNFactor: Multi-task real robot learning with generalizable neural feature fields," in *Conference on Robot Learning*, 2023, pp. 284–301.

[9] G. Yan, Y.-H. Wu, and X. Wang, "DNAct: Diffusion guided multi-task 3D policy learning," *arXiv preprint arXiv:2403.04115*, 2024.

[10] A. Vaswani, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[11] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," in *Advances in Neural Information Processing Systems*, vol. 35, pp. 22955–22968, 2022.

[12] A. Jaegle *et al.*, "Perceiver IO: A general architecture for structured inputs & outputs," *arXiv preprint arXiv:2107.14795*, 2021.

[13] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[14] C. Chi *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, 2023.

[15] K. Tian *et al.*, "Visual autoregressive modeling: Scalable image generation via next-scale prediction," *arXiv preprint arXiv:2404.02905*, 2024.

[16] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," in *Advances in Neural Information Processing Systems*, vol. 34, pp. 8780–8794, 2021.

[17] J. Ho *et al.*, "Cascaded diffusion models for high fidelity image generation," *Journal of Machine Learning Research*, vol. 23, no. 47, pp. 1–33, 2022.

[18] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[19] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12873–12883.

[20] R. Rombach *et al.*, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10684–10695.

[21] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*, 2021, pp. 8748–8763.

[22] S. Klemm *et al.*, "RRT*-Connect: Faster, asymptotically optimal motion planning," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2015, pp. 1670–1677.

[23] H. Lin *et al.*, "Efficient neural radiance fields for interactive free-viewpoint video," in *SIGGRAPH Asia 2022 Conference Papers*, 2022, pp. 1–9.

[24] Y. You *et al.*, "Large batch optimization for deep learning: Training BERT in 76 minutes," *arXiv preprint arXiv:1904.00962*, 2019.

[25] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "RLBench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.