Unifying Diffusion Models with Action Detection Transformers for Multi-task Robotic Manipulation

Anonymous Author(s)

Abstract: We present ChainedDiffuser, a policy architecture that unifies transformer-based end-effector action prediction and diffusion-based trajectory generation for learning multimodal multi-task robotic manipulation from demonstrations. Our model sets a new record on established manipulation benchmarks across a variety of settings, significantly outperforming all prior state-of-the-art approaches. Our main innovation is to use a global transformer-based action predictor to predict actions at keyframes, a task that requires multimodal semantic scene understanding, and to use a local trajectory diffuser to predict trajectory segments that connect predicted macro-actions. ChainedDiffuser outperforms both state-of-the-art macro-action prediction models that use motion planners for trajectory prediction, and trajectory diffusion policies that do not predict keyframe macro-actions. We conduct experiments in both simulated and real-world environments and demonstrate ChainedDiffuser’s ability in solving a wide range of manipulation tasks involving interactions with diverse objects.

Keywords: Manipulation, Imitation Learning, Transformers, Diffusion Models

1 Introduction

While learning manipulation policies from demonstrations is a supervised learning problem, the multimodality and diversity of action trajectories poses significant challenges to machine learning methods. Some tasks, such as placing a cup in a cabinet, can be handled by a policy that provides only a desired goal pose for the cup [1, 2, 3], while others, such as wiping off dirt on the floor, necessitate the policy to generate a continuous action trajectory [4, 5, 6] for the grasped mop. One line of manipulation learning methods model complete action trajectories present in demonstrations. These methods either reactively map vision and language input to dense temporal actions [7, 8, 9, 10, 11], or model the input-action compatibility using energy-based models [12, 13, 14, 15]. Despite recent progress, these methods may struggle with multimodal action trajectory distributions, or experience issues related to training stabilities [16, 15, 17]. Building on successes in diffusion models [18, 19, 20], a recent line of work proposes to train diffusion-based policies [16, 5, 21] for generating action trajectories. These approaches have demonstrated stable training behavior and impressive capacity in capturing multimodal action trajectory distributions. Yet, they haven’t yet been tested on handling 3D visual scene configurations and long-horizon language-conditioned multi-task learning.

Another line of works re-frames the problem of action trajectory prediction to predicting a sequence of discrete end-effector actions [1, 22, 14]. This paradigm extracts keyframes from continuous demonstrations and predicts end-effector actions in these keyframes, often using detection architectures from computer vision research [2, 23, 3]. Subsequently, a low-level path planner connects these predicted macro-actions, and returns full trajectories that adhere to both environmental and task constraints. Leveraging recent advances in attention-based architectures [24], a number of methods...
extend keyframe action prediction to 6-DoF language-instructed manipulation tasks [2, 3, 23, 25]. However, the assumptions behind keyframe prediction hinder its applicability to manipulation tasks that extend beyond discrete pick-and-place actions (e.g., wiping a table, opening a door while respecting the kinematic constraints, etc.), which can only be solved via continuous interactions with the environment. Moreover, the dependence on low-level path planning further restricts these methods’ capability: while a range of tasks need collision-free trajectories, other tasks, such as object pushing [25, 3, 26], necessitate that the motion planner disregards collision avoidance. Although supervision for this additional reasoning is readily available in simulated datasets [27] and can be learned by the policy, real-world human demonstrations typically lack such data, not to mention that collision-free motion planning in the real world requires accurate state estimation of both the manipulated object and the surrounding environment, which presents its own challenges.

In light of the above, we present ChainedDiffuser, a neural architecture that unifies the two aforementioned paradigms, combining their strengths while addressing their respective limitations. ChainedDiffuser is a multi-task policy learner that takes as input both visual signals and a language instruction and outputs temporally dense end-effector actions. At a coarse level, it predicts macro-step end-effector actions (which we will call *macro-actions*), a high-level task that requires global comprehension of the visual environment and the language instruction, with a global transformer-based action predictor. Then, a low-level trajectory diffuser generates local trajectory segments to connect the predicted macro-actions. In comparison to transformer-based macro-step prediction methods [2, 3, 23, 25], our model predicts smooth trajectories to accommodate tasks that require continuous interactions and collision-free actions. In comparison to diffusion-only trajectory generation methods [16, 17, 5, 21], our hierarchical approach handles long-horizon tasks in a more structured manner, and allow different modules to concentrate on the tasks at which they excel.

We test ChainedDiffuser on RLBench [27], a well-established benchmark for language-conditioned manipulation learning from demonstrations. We evaluate our model across a variety of scenarios, including single-task, multi-task, and multi-variation setups studied in previous literature [23, 25, 3]. ChainedDiffuser achieves a success rate over 95% on 43 out of 74 RLBench tasks, and significantly outperforms all prior state-of-the-art methods. We further show ChainedDiffuser outperforms ablative versions of our model that do not predict macro-actions or use regression or planners for keyframe-to-keyframe trajectory prediction. Furthermore, we validate our model in real-world scenarios with a number of long-horizon manipulation tasks, using a handful of human demonstrations for training. Our code will be made publicly available upon publication.

2 Related Work

**Learning from Demonstrations** [28, 29] has been a common paradigm for robotics but requires demonstration data collection in the real world [7, 30, 31] or simulation [27, 32, 33]. To improve data efficiency, several approaches learn the policy on top of pre-trained visual representations that exploit large vision-only datasets [34, 35, 36, 37, 38, 39]. Orthogonal to this, other approaches abstract every task as a sequence of subgoals, expressed as pick-and-place primitives [1, 2] or keyframes [40, 41]. In this case, hand-designed low-level controllers are employed to plan the end-effector’s motion between intermediate subgoals. While data-efficient, this abstraction does not generalize adequately to scenarios where only few specific trajectories that respect all physical constraints are valid [42], such as manipulation of deformable [43] or articulated [44] objects, or motion through obstacles in a cluttered environment [45]. As a result, recent works resort to semi-manual cost specification for each additional constraint (e.g., collision avoidance, trajectory smoothness [5]), that often demand prior knowledge of the environment/objects and therefore cannot scale to novel environments and more complex tasks. Closer to our approach, James and Abbeel [42] learn to score trajectories proposed by either hand-designed or learning-based planners. Instead, we employ diffusion model to generate trajectories from randomly sampled noises.

**Transformers for Robotics**: Following their success in natural language processing [24, 46, 47] and computer vision [48, 49], numerous recent works use Transformer-based architectures for robotics
and control [50, 51, 7, 52, 53]. One main motivation is the flexibility of attention for long-horizon prediction when combining information from multiple sensory streams, such as visual observations and language instructions [54, 23]. Most related to ours is the stream of multi-tasking Transformer-based models, that are trained on diverse datasets to achieve higher in-distribution [3, 25] or out-of-distribution generalization [7, 55, 56, 57]. Our model comprises of two attention-based modules, one for macro-step action prediction and one for local trajectory optimization, that can leverage different input modalities and operate over different abstractions.

**Diffusion Models** [58, 20, 18, 19] learn to approximate the data distribution through an iterative denoising process, and have shown impressive results on both unconditional and conditional image generation [59, 60, 61, 62]. In the field of robotics, diffusion models find applications on planning [17, 63, 64], scene re-arrangement [65, 66], controllable motion optimization [67, 68], video generation [69] and imitation learning [16, 21]. Their main advantage is that they can better capture the action trajectory distribution compared to previous generative models. Recent works use diffusion model to predict complete trajectories, often auto-regressively [16, 70]. Instead, we use diffusion models to generate local trajectories that are chained together with macro-actions.

### 3 ChainedDiffuser

#### 3.1 Overview

The architecture of ChainedDiffuser is illustrated in Figure 1. It is a model for robotic manipulation that combines macro-action prediction with cross- macro-action conditional trajectory diffusion. It uses as input visual observations of the environment and a natural language description \( l \) of the task as input. At each step, ChainedDiffuser predicts a macro-action \( \hat{a}_t \) using a global policy \( \pi_{\text{global}} \), and then feeds \( \hat{a}_t \) together with its current end-effector state \( q_t \) to a low-level local trajectory generator \( \pi_{\text{local}}(q_t, \hat{a}_t) \) to generate dense micro-actions connecting \( q_t \) and \( \hat{a}_t \), as shown in Figure 1. Both \( \hat{a}_t \) and \( q_t \) share the same space \( A = \{a_{\text{pos}}, a_{\text{rot}}, a_{\text{grip}}\} \), consisting of the end-effector’s 3D position \( a_{\text{pos}} \), rotation \( a_{\text{rot}} \) represented as a 4D quaternion, and a binary flag \( a_{\text{grip}} \) indicating whether the gripper is open. For each task, we assume access to a dataset \( D = \{\zeta_1, \zeta_2, \ldots, \zeta_m\} \) of \( m \) expert demonstrations, where \( \zeta_i \) contains the language instructions \( l \), visual observations \( o \) and end-effector states \( q_t \) for all timesteps in the demonstration.

**Input Encoding** ChainedDiffuser operates in a 3D space to achieve robustness across changing camera viewpoints – an important advantage over prior 2D methods which assume fixed camera viewpoints [23, 25, 16]. Compared to prior robotic architectures which rely on voxel-based 3D representation (e.g., [3, 41]), ChainedDiffuser employs a point-based representation, facilitates sparse computation and circumvents precision loss during voxelization. ChainedDiffuser uses a frozen CLIP [71] to encode both the language instruction \( l \) and the RGB images \( q_t \) into a set of language and visual feature tokens respectively. Then, it uses the depth channel information to unproject the 2D image feature tokens into a 3D feature cloud (Figure 1(c)), where each visual token has 2D appearance information and 3D positional information. We also encode the proprioception information \( q_t \) with a simple MLP.

#### 3.2 Macro-Action Detector

Our macro-action detector \( \pi_{\text{global}} \) is based on Act3D [72], our concurrent CoRL submission (included in Appendix for reviewers’ reference, per CoRL’s policy). Act3D is a point-based transformer that casts end-effector action prediction as a detection problem and detects 6-DoF actions in 3D space. Act3D samples iteratively 3D point candidate grids and featurizes them using relative position attentions to the lifted 3D feature cloud. We include its main pipeline here for completeness: Act3D instantiates a trainable query token \( Z_{\text{query}} \) and uses it to score a pool of \( N \) point candidates \( \{P_i = \{x_i, y_i, z_i\}\}_{i=1}^{N} \) in the scene to select a position for next macro-action. The point candidates are first uniformly sampled within the robot’s empty workspace and only contain 3D positional information and a trainable feature embedding \( Z_{\text{point}} \). We then contextualize both the query token and the point candidates, by letting them individually attend to the concatenation (across the sequence dimension) of language tokens \( Z_{\text{ins}} \), visual feature tokens \( Z_{\text{vis}} \), and proprioception token \( Z_{\text{robot}} \).
Figure 1: ChainedDiffuser is a language-conditioned multi-task policy trained to imitate action trajectories from demonstrations. Here we illustrate the method for the task “wiping the beans on a table”. Note that the feature cloud in c) consists of encoded feature tokens, but are visualized in RGB for illustration purposes.

(Figure 1(e)):

\[ \hat{Z}_{\text{query}} = \text{Attn}(Z_{\text{query}}, \langle Z_{\text{ins}}, Z_{\text{vis}}, Z_{\text{robot}} \rangle), \quad \hat{Z}_{\text{point}} = \text{Attn}(Z_{\text{point}}, \langle Z_{\text{ins}}, Z_{\text{vis}}, Z_{\text{robot}} \rangle), \]

where \( \text{Attn}(x, y) \) is an attention operation [24, 73] where the queries are formed from \( x \), the keys and values from \( y \). Act3D uses relative 3D positional encodings proposed in [74, 75] to incorporate translational invariance. After this contextualization step, the query token and the point candidates have captured the task and scene information. We take the dot product of the contextualized query embedding with all point candidates and select the best-matching point candidate for the position of the predicted macro-action, and regress the rotation and gripper open flag with a simple MLP on top of the query.

### 3.3 Local Trajectory Diffuser

Once we obtain the macro-action \( \hat{a}_t \) for the current step \( t \), we call upon our diffusion-based local trajectory generator to fill up the gap in-between with micro-actions. We model such trajectory generation as a denoising process [20, 16, 5]: we start with drawing a sequence of \( S \) random Gaussian samples \( \{ x^k_s \}_{s=1}^S \) in the normalized SE(3) space, and then perform \( K \) denoising iterations to transform the noisy trajectories to a sequence of noise-free waypoints \( \{ x^s_0 \}_{s=1}^S \). Each denoising iteration is described by:

\[ x^{k-1}_s = \lambda_k(x^k_s - \gamma_k \epsilon_k(x^k_s, k)) + \mathcal{N}(0, \sigma_k^2 I), \quad 1 \leq s \leq S \tag{1} \]

where \( \epsilon_k \) is the noise prediction network, \( k \) the denoising step, \( \mathcal{N}(0, \sigma_k^2 I) \) the Gaussian noise added at each iteration, and \( \lambda_k, \gamma_k, \sigma_k \) are scalar noise schedule functions dependent on \( k \) (Appendix 6.1). The noise prediction network (Figure 1 (f)) is also an attention-based model that absorbs similar input as the macro-action selector does, i.e., the language instruction \( l \), RGB-D observations \( o_t \) and current end-effector state \( q_t \), but additionally conditions on the goal macro-action \( \hat{a}_t \) and the denoising timestep \( k \). The language tokens \( Z_{\text{ins}}, \) visual tokens \( Z_{\text{vis}}, \) and current end-effector state \( Z_{\text{robot}} \) are featurized similarly to the Macro-Action Selector. We use an MLP to encode the goal macro-action into \( Z_{\text{macro}} = \text{MLP}(\hat{a}_t) \). We encode the denoising timestep into \( Z_{\text{time}} \) using sinusoidal
positional embeddings [24], and encode the the sampled noise using an MLP into a sequence of
tokens $Z^k_s$. We let this sequence iteratively cross-attend to all encoded inputs and self-attend:

$$
\hat{Z}^k_s = \text{Attn}(Z^k_s, \langle Z_{\text{ins}}, Z_{\text{vis}}, Z_{\text{robot}}, Z_{\text{macro}}, Z_{\text{time}} \rangle) 
$$

(2)

Again, we use relative positional embeddings to encode all tokens’ spatial positions. For the tra-
jectory noise tokens, we additionally encode each sample’s temporal position $s$ using sinusoidal
positional embeddings. These are added to the respective noise tokens $Z^k_s$. The contextualized
noise sample is then fed into another MLP for noise regression:

$$
\epsilon_{\theta}(x^k_s, k) = \text{MLP}(Z^k_s)
$$

(4)

After $K$ denoising steps by substituting Equation 4 into 1, we convert the denoised samples back to
the actual micro-actions by unnormalizing them: $a_{t-1+s} = \text{Unnormalize}(x^0_s), \quad 1 \leq s \leq S.$

3.4 Implementation and Training Details

ChainedDiffuser takes as input $m$ multi-view RGB-D images of the scene. For experiments in
simulation, we use $m = 3$ (left, right, wrist) or $m = 4$ (with an additional front view), depending
on the settings of the baselines we compare with. For real-world experiments, we use $k = 1$, with a
single front-view camera. Each RGB-D image is $256 \times 256$ and is encoded to $64 \times 64$ visual tokens
with CLIP’s ResNet50 visual encoder [71]. The demonstration data contains end-effector states for
all timesteps. In order to extract macro-actions to supervise the action selection transformer, we use
a simple heuristic following previous literature [3, 23, 25]: a timestep is considered to be a keyframe
containing macro-action if the gripper opens or closes, or if the robot arm is not moving (when all
joint velocities approach zero). All dense actions present in the demonstration are used to supervise
the local trajectory diffuser. We resample the dense trajectories between extracted macro-actions to
a trajectory of fixed length $S = 50$. We found in practice, denoising fixed number of micro-actions
leads to more stable training, and works better than learning variable-length trajectory diffusion
with predicted trajectory length. We train both the action detector and the trajectory diffuser jointly,
using a cross-entropy (CE) loss to supervise the point candidate selection by predicting a probability
distribution $q$ over all point candidates in the pool, and mean-squared error (MSE) losses to supervise
quaternion, gripper opening and trajectory noise regression:

$$
\mathcal{L} = \frac{1}{|D| |\zeta|} \sum_{\zeta \in D} \sum_{i \in \zeta} \left[ \text{CE}(q(\{P_i\}^N), q^*(\{P_i\}^N)) + \text{MSE}(\hat{a}_i, a_i^*) + \sum_{t=i}^{t+S-1} \text{MSE}(a_t, a_t^*) \right], \quad (5)
$$

where * indicates predicted value, and $\hat{a}_i = \langle \hat{a}_{\text{rot}}, \hat{a}_{\text{grip}} \rangle$. We use a batch size of $B = 24$ and
AdamW [76] optimizer with a learning rate of $1e-4$ for all our experiments. Our single-task model
is trained for 1 day on one A100 GPU, and multi-task model is trained for 5 days on 4 A100 GPUs.

4 Experiments

We test ChainedDiffuser in both single- and multi-task learning settings, using both simulated
and real-world environments. Our experiments aim to answer the following questions: 1) How
does ChainedDiffuser compare to previous SOTA 2D and 3D manipulation methods on established
benchmarks? 2) How much performance boost can be achieved by delegating local trajectory gener-
atution to trajectory diffusers? 3) Is macro-action prediction helpful in guiding trajectory generation?
4) Does ChainedDiffuser work in real-world settings where only a single camera and limited number
of demonstrations are available?

4.1 Simulation Experiments

We run experiments in simulation using RL Bench [27], a widely adopted manipulation benchmark
with diverse tasks concerning interactions with a range of different objects (See Figure 2). We
follow the same setting used in prior works [23, 25, 3], where each task has multiple variations and contains 100 demonstrations with a set of language instructions. We report success rates on all evaluated averaged over 100 unseen test set episodes for each task. For all the baselines, we use the official numbers reported in their papers. Please see our Appendix 6.3 for more details on the simulation setup.
We train single-task ChainedDiffuser on 74 tasks in RLBench, group them into 9 categories according to their challenges following [23, 25, 77], and report success rate for each group and success rate averaged over all tasks in Table 1. ChainedDiffuser consistently achieves better performance than prior methods on this setting on all task categories, and outperforms Act3D, the previous state-of-the-art method under this setting. Compared to Act3D w/o Col. Avoid. ChainedDiffuser shows a 9% boost on average, and still maintains a 4 point boost over Act3D which requires additional collision avoidance supervision, demonstrating the advantage of incorporating local trajectory predictions instead of hand-designed planners.

**Multi-task evaluation** We train a single language-conditioned policy across all tasks and compare against the following baselines: 1) Auto-λ [77], HiveFormer [25] and InstructRL [23] on a 10-task learning setting, same as the one evaluated in these papers (“HiveFormer setting”); 2) with Act3D and PerAct [3], a multi-task policy that operates in voxelized 3D space and detects end-effector action with global self-attention built on top of Perceiver IO [78], on a 18-task 249-variation setting proposed by PerAct (“PerAct setting”). Each task evaluates generalization capabilities over varying object colors, shapes, sizes, spatial configurations, and categories. We report the comparison in Table 2 and 3. On the 10-task HiveFormer Setting, ChainedDiffuser consistently achieves over 90% success rate on all the tasks with a single policy. Since these 10 tasks adopt relatively simple task settings, prior methods are already performing well and ChainedDiffuser shows a small gain on average. On the 18-task PerAct Setting, our unified model outperforms PerAct by 23% on average. Compared to Act3D, ChainedDiffuser achieves similar performance with a minor gain. This is because the tasks in the PerAct settings are designed to be solvable by keyframe end-effector prediction methods alone, which we will discuss in the next experiment setting. The multi-task evaluation results here suggests that our multi-task local trajectory diffuser is able to generate reasonable trajectories, and achieve competitive results when replacing motion planners on tasks where motion planning suffices.

**On the importance of unification** In the previous experiments, ChainedDiffuser outperforms prior state-of-the-art methods by a big margin, but leads to a small gain over Act3D, the best macro-step action prediction method so far. We observe that this is because most of the tasks in RL Bench can be reasonably solved with only discrete action prediction and low-level motion planners, and the tasks in the PerAct Setting are manually selected by the authors and do not contain tasks that require continuous interactions, which is not the case in many household tasks. In order to better illustrate the importance of unifying local trajectory generation with macro-step action prediction, we iterate over the tasks in RL Bench and identify 10 challenging tasks that requires continuous interaction with the environment, such as wipe desk where a wiping trajectory is needed to remove all the dirt, and open fridge where a local trajectory needs to adhere to the kinematic constraint when the robot is grasping the door handle. Under this setting, we compare with 1) Macro-Action Only,
same as Act3D, the de-facto SOTA model that relies on low-level motion planner to connect macro-actions; 2) Diffusion Only, which is ChainedDiffuser without the macro-action detector, making it a trajectory generation model based on diffusion alone, similar to Diffusion Policy [16]; 3) Regression, which shares most of the components with ChainedDiffuser except replacing the local trajectory diffuser by a direct regression layer. We report the numbers in Table 4. On this set of challenging tasks that previous macro-step prediction methods struggle with, ChainedDiffuser gives a significant boost of 60% on average. Compared with Diffusion Only model, ChainedDiffuser also shows a 49% improvement, demonstrating that delegating global macro-action prediction to high-level detector to guide local trajectory diffusion will lead to significantly better performance. The number also shows that modeling trajectory generation as a multi-step denoising process is advantageous over regression-based model, which aligns with conclusions from previous literature [16].

4.2 Real-world Experiments

We conduct experiments with a real-world setup, using a Franka Emika Panda robot with a parallel-jaw gripper. We use a single Azure Kinect camera to collect front-view RGB-D image input. See Appendix 6.4 for more details on our hardware and data collection setup. We design 7 tasks that involve multi-step actions and continuous interactions with the scene (5 are shown in Figure 2), collected 10 – 20 demos for each tasks, and train a multi-task ChainedDiffuser for real-world deployment. We refer the reader to our supplementary video for qualitative executions of the robot. We evaluate it on 10 episodes for each task, and report success rates in Table 5. ChainedDiffuser is able to perform reasonably well on most of the tasks, even for tasks with multiple action modes and skills. The most common failure case is caused by noisy depth image: we leverage point selection for macro-action prediction, which would suffer from incorrect depth estimation in the real world. This could potentially be resolved by more accurate camera calibration with a multi-view camera setup and learning to recover from noisy input, which we leave as our future work.

4.3 Limitations

Our method currently has the following limitations: 1) It leverages position information encoded in the input depth images, which is a major bottleneck in real-world when the input signal is noisy. 2) Our trajectory diffuser is conditioned on end-effector poses in SE(3) space. It would be ideal to extend it to full joint configuration space for more flexible trajectory prediction. 3) Our model performs closed-loop control on the macro-action level, which restricts its flexibility in highly dynamic environments. That said, our framework can be easily extended with closed-loop re-planning at the micro-action level, making the policy more robust to environment dynamics, which we leave as our future work.

5 Conclusion

We presented ChainedDiffuser, a neural architecture for 6-DoF language-conditioned manipulation policy learning. We show our approach achieves competitive performance on various task settings, in both simulation and in the real-world. Our experiments demonstrate that by unifying both transformer-based macro-action detection and diffusion-based trajectory generation, ChainedDiffuser takes the best of both families and addresses their respective limitations. We set a new state-of-the-art in RLbench, and dramatically improves performance on tasks that involves articulated objects and contact-rich tasks, which previous works that relied on hand designed planners typically struggled with. We further show that ChainedDiffuser outperforms both keyframe prediction methods and trajectory diffusion alone, which justifies their unification in our framework.
References


