Accelerating Residual Reinforcement Learning with Uncertainty Estimation

Lakshita Dodeja*, Karl Schmeckpeper†, Shivam Vats*, Thomas Weng† and Stefanie Tellex*
*Department of Computer Science, Brown University, Providence, RI
†RAI Institute, Cambridge, MA

***Abstract*—Residual Reinforcement Learning (RL) is a popular approach for adapting pretrained policies by learning a lightweight residual policy to provide corrective actions. While Residual RL is more sample-efficient compared to finetuning the entire base policy, existing methods struggle with sparse rewards and are designed for deterministic base policies. We propose two improvements to Residual RL that further enhance its sample efficiency and make it suitable for stochastic base policies. First, we leverage uncertainty estimates of the base policy to focus exploration on regions in which the base policy is not confident. Second, we propose a simple modification to off-policy residual learning that allows it to observe base actions and better handle stochastic base policies. We evaluate our method with both Gaussian-based and Diffusion-based stochastic base policies on tasks from Robosuite and D4RL, and compare against state-of-the-art finetuning methods, demo-augmented RL methods, and other residual RL methods. Our algorithm significantly outperforms existing baselines in a variety of difficult manipulation environments.**

I. Introduction

Residual Reinforcement Learning is popular in robotics for adapting pretrained robot policies using a residual agent that learns to maximize reward through environment interactions [26, 14]. Directly fine-tuning the pretrained policy is often computationally expensive, especially in the case of policies with a large number of parameters [7, 4] and is prone to instability [18]. In contrast, Residual RL provides an efficient alternative to refine the base policy with minimal additional computation. This residual correction enables the agent to make targeted improvements, regardless of whether the base policy is model-based or model-free.

Despite the promise of Residual RL, existing algorithms suffer from unconstrained exploration, often requiring extensive online interaction and dense reward shaping to achieve meaningful improvements [16, 29]. Furthermore, recent advancements in imitation learning leverage highly effective stochastic policies: Gaussian mixture model-based policies [17] and Diffusion policies [7] excel at modeling complex, multi-modal distributions. Unfortunately, original Residual RL algorithms [26, 14] are not suitable for such stochastic policies as they implicitly assume that the underlying base policy is deterministic.

In this paper, we address these limitations by proposing two improvements to Residual RL that increase sample complexity and are suitable for stochastic policies. First, we leverage uncertainty estimates from the base policy to guide the exploration of the residual policy. Our key insight is that regions where the base policy is confident require minimal exploration by the residual agent, allowing it to focus exploration on areas of high uncertainty. This targeted exploration significantly improves the sample efficiency of residual learning. Second, existing off-policy Residual RL algorithms learn the $Q(s, a_r)$ function only for the residual action a_r, implicitly assuming that the base policy's action can be inferred from the state s. However, this is insufficient when dealing with stochastic base policies, since they can take different actions from the same state. In the stochastic setting, the Residual RL agent is unable to infer the base action, making it difficult to effectively learn a good residual action. Some works have used learned bottleneck features of the base policy as a prior for residual learning [1] while others augmented the observed state with the base action for on-policy learning [3]. We propose an asymmetric actor-critic RL approach, in which the critic learns the Q function for the fully observed action executed in the environment, comprising both the base action and the residual action, while the actor learns partial residual actions only. This formulation ensures that information about the stochastic base actions is available to the Q function while also making the learning invariant to the split between the residual and base action.

Fig. 1: We test our proposed approach on the *Lift*, *Can*, and *Square* tasks from Robosuite [17] and the *Franka Kitchen* Task from D4RL [8].

We evaluate our approach on a variety of manipulation tasks from Robosuite [17] and Franka Kitchen tasks from D4RL [8]. We also test our approach with both Gaussian Mixture Model-based and Diffusion-based base policies. Finally, we compare our approach with state-of-the-art finetuning methods [21], demo-augmented RL method [13], and other Residual RL methods [28, 26, 14]. Our proposed approach is able to outperform or is comparable to other baselines in all tasks. We also perform several ablation studies to test various aspects of our algorithm.

Our proposed approach is visualized in Figure 2 with the main contributions summarized as follows:

1) We present a novel algorithm to mitigate the exploration problem of Residual Reinforcement Learning using un-

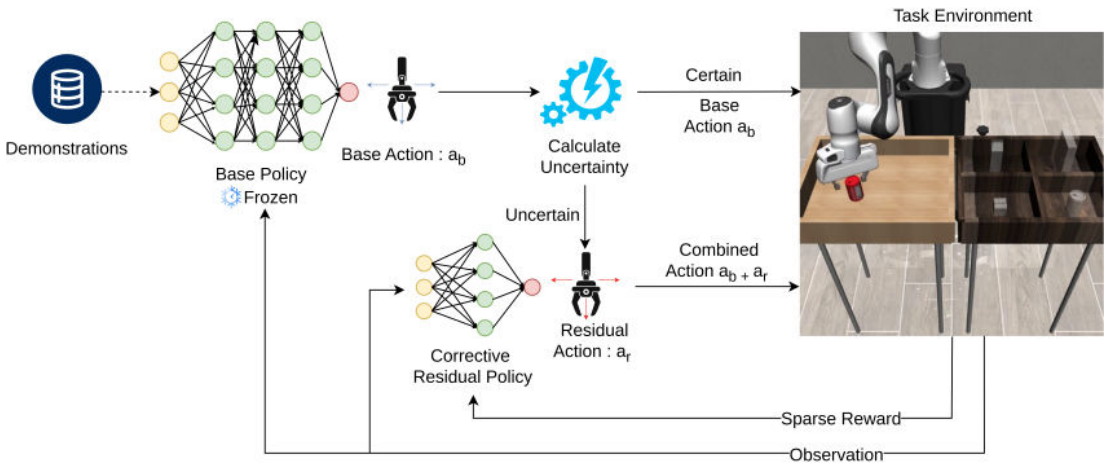

Fig. 2: We propose two improvements to accelerate Residual RL : 1) We use uncertainty metric to constrain the exploration around base policy by using the base action directly if the base policy is certain and adding the residual action when the base policy is uncertain. 2) We modify off policy critic to learn the Q function for the combined action.

certainty estimates

2) We modify off-policy Residual Reinforcement Learning to work with stochastic base policies by providing the combined base and residual action to the critic and sampling the residual action from the actor.
3) We validate our method on robotic manipulation tasks from different simulators and against several baselines.

II. Related Work

Imitation Learning and Residual RL. Several works have explored the integration of base policies trained using Imitation Learning (IL) with Residual RL. [23] applied Residual RL to insertion tasks, utilizing both a hand-designed controller as the base policy and incorporating an auxiliary behavior cloning (BC) loss during RL training, a technique first proposed in [19]. FISH [12] employs a non-parametric base policy alongside a residual policy that uses optimal transport matching against offline demonstrations as the reward. BeT [25] introduces a residual action corrector that refines continuous actions on top of a discretized imitation policy. IBRL [13] does not directly use Residual RL but it bootstraps an RL policy from an IL policy by utilizing IL actions as alternative proposals for both online exploration and critic updates. Closest to our method is Policy Decorator [28] which learns bounded residual actions using controlled exploration. Unlike Policy Decorator, which uniformly samples action from the base policy and residual policy, we use uncertainty estimates of the base policy to decide when to learn corrective residual actions.

Residual RL for Stochastic Base Policies. Residual RL, first introduced for robotics in [14] and [26], learns a corrective residual policy over a base controller, which can be either hand-designed or derived from model-predictive control. Importantly, these methods assume a deterministic base controller, as the residual policy is not conditioned on the base action. However, current state-of-the-art imitation learning algorithms like Diffusion policy [7] and GMM-based policies [17] are non-deterministic making the original Residual RL formulation insufficient due to the lack of information about the base policy. [16, 29] introduce noise in the base action to enhance robustness and induce stochasticity. Other works inform the residual learning about the base policy by conditioning the residual learning on the learned bottleneck features of the base policy [1], and incorporating the base action in the observed state to inform the residual policy for on-policy learning [3]. In contrast, our work modifies off-policy RL algorithm to learn the Q function for the combined action taken in the environment (i.e. the sum of base and residual action) while the actor still uses the same Q function to select a residual action. Therefore, our Residual RL formulation can handle the stochasticity of the base policies by making the base action observable to the critic while also being invariant to the split between the base action and residual action. Policy Decorator [28] also uses the combined action as an input to their critic for Residual RL, but we provide precise pseudo code for changing the SAC algorithm as well as a thorough quantitative evaluation with ablations to show that it significantly improves performance.

Uncertainty estimates in Imitation Learning. Uncertainty estimation plays a crucial role in improving the reliability and robustness of machine learning models, particularly in decision-making and RL tasks. Various approaches have been proposed to quantify uncertainty, including distance-based techniques [20, 27], Diffusion-based techniques [6, 15], ensemble-based techniques [2, 24, 9] and learning another model to estimate uncertainty [22, 5, 10]. Suh et al. [27] proposes a method that measures the distance of a given input to the training data distribution. This approach assumes that samples farther from the training distribution exhibit higher uncertainty, making it particularly useful for detecting out-of-distribution (OOD) inputs and improving model generalization. Lee and Kuo [15] uses the loss function of

a Diffusion model to estimate uncertainty, where higher loss values indicate greater uncertainty. Our algorithm is agnostic to the uncertainty quantification method, and we perform ablations with different uncertainty metrics.

III. Uncertainty aware Residual RL for stochastic policies

The problem statement for our method is defined in Sec. III-A. We describe how to incorporate Uncertainty Estimates in Residual RL in Sec. III-B. Finally, we describe our modified off-policy RL algorithm in Sec. III-C.

A. Problem Statement

In Residual RL, we assume that we have a suboptimal base policy π_b, either model-based or model-free. The objective is to learn a lightweight residual policy π_r, on top of the base policy that gives a corrective action a_r to produce a more accurate and robust policy. Residual RL transforms the original markov decision process (MDP) formulation $M =< S, A, R, T, \gamma >$ to the residual MDP (RMDP) formulation $M_r =< S, A_r, R, T_r, \gamma >$ [26]. S is the set of states, A_r is the set of residual actions, R is the reward received for taking action $a_r \in A_r$ in state $s \in S$, T_r is the probability of taking action a_r in state s and ending up in a new state s' and γ is the discount factor. The residual transition function can be converted back to original transition function as follows :

$$T_r(s, a_r, s') = T(s, \pi_b(s) + a_r, s') \tag{1}$$

B. Uncertainty aware Residual RL

Prior works in Residual RL suffer from unrestrained exploration as they learn corrective residual actions uniformly over the entire state space. Our key insight is to improve exploration by focusing residual learning on regions in which the base policy is not confident. We propose using the uncertainty of the base policy to decide when to learn a residual action for the base policy. If the base policy is certain about its action for the current state we directly use the action from base policy a_b to step in the environment and instead use a corrective residual action when the base policy is uncertain. Our proposed approach is agnostic to the uncertainty quantification method and we further test our method with two metrics, namely distance-to-data and ensemble variance. Distance-to-data has been used to calibrate the uncertainty of a model by measuring how out-of-distribution the current state is from an existing dataset [27]. For a dataset D with each datapoint having F features, we can estimate uncertainty using the minimum L2 distance of the current state s to all points $d \in D$:

$$\text{uncertainty}_d(s) = \min_{d \in D} \sqrt{\sum_{i=1}^{F} (d_i - s_i)^2} \tag{2}$$

Another popular approach to calibrate uncertainty is by measuring the variance in predicted actions amongst an ensemble of policies. For an ensemble of N base policies $\pi_b \in \pi_B$, the ensemble uncertainty can be defined as -

$$\text{uncertainty}_e(s) = \frac{1}{N} \sum_{b=1}^{N} \left(\pi_b(a \mid s) - \frac{1}{N} \sum_{i=1}^{N} \pi_i(a \mid s) \right)^2 \tag{3}$$

Furthermore, we use an uncertainty threshold τ to measure the confidence of the base policy. This can be formulated as

$$a_{\text{taken}} = \begin{cases} a_{\text{b}} & \text{if uncertainty} < \tau \\ a_{\text{b}} + a_{\text{r}} & \text{otherwise} \end{cases} \tag{4}$$

As learning progresses, we decay this uncertainty threshold τ exponentially from a maximum uncertainty threshold value U according to the following equation:

$$\tau = U * e^{\frac{-\text{step}}{\text{decay rate}}} \tag{5}$$

The uncertainty threshold τ ultimately decays to 0 to let the residual policy take over. We perform ablations for different decaying strategies in Sec. IV-E1.

C. Optimizing Residual RL for Stochastic Policies

The original Residual RL algorithms are formulated to learn only using the partial residual actions, operating under the assumption that the underlying base policy is deterministic and can be implicitly inferred. Therefore, it learns the Q function for only the partial residual action which is different from the action taken in the environment. Incorporating stochastic policies into the residual transition function T_r can make it much harder to learn as they are noisier in their predictions and hence difficult to model. Thus, we suggest using Eq 1 to retreat back to the original MDP formulation. Consequently, the Q function is learned for the combined action ($a_c = a_b + a_r$) which is the actual action used during environment interaction. Previously, ResiP [3] proposed augmenting the observed state with base action to provide the missing information for on-policy RL. We instead propose learning the critic for the combined action ($a_c = a_b + a_r$) for off-policy RL to provide the necessary information about the base policy to the Q function while also making it invariant to the split between residual action and the base action. Specifically, we modify the soft actor-critic [11] algorithm in the following ways (changes marked in green). Initially, we store both the base action and the combined action in the replay buffer. While computing the target values, we should add the base action to the residual action sampled from the actor as follows -

$$\begin{aligned} y(r, s', d) = r + \gamma(1 - d) * \\ (\min_{i=1,2} Q_{\phi_i'}(s', a_r' + a_b') - \alpha \log \pi_r(a_r' | s')), \\ a_r' \sim \pi_r(\cdot | s') \end{aligned} \tag{6}$$

While updating the Q function we should use the combined action stored in the replay buffer -

$$J_Q(\phi_i) = \mathbb{E}\left[(Q_{\phi_i}(s, a_c) - y(r, s', d))^2 \right], \quad i = 1, 2 \tag{7}$$

When updating the actor, we can again add the base action to get the Q value -

$$J_\pi(\theta) = \mathbb{E}\left[Q_{\phi_i}(s, a_r + a_b) - \alpha \log \pi_r(a_r|s)\right], \quad i = 1, 2, \quad a_r \sim \pi_r(\cdot|s) \tag{8}$$

The complete algorithm is described in Alg. 1 with the proposed changes in SAC marked in green.

Algorithm 1 Uncertainty aware Residual RL

Initialize parameters of residual policy π_r, Q-functions Q_{ϕ_1}, Q_{ϕ_2}, and temperature α
Initialize target Q-function parameters $\phi_1' \leftarrow \phi_1$, $\phi_2' \leftarrow \phi_2$
Initialize base policy π_b
for each environment step **do**
  Sample residual action from actor
    $a_r \sim \pi_r(s_t)$
  Sample base action from base policy
    $a_b \sim \pi_b(s_t)$
  Calculate uncertainty threshold τ for the base policy, Eq. 5

  Calculate the uncertainty in base policy, Eq. 2
  Select the action to be taken in the environment, Eq. 4
  Observe next state s_{t+1} and reward r_t
  Store $(s_t, a_c, a_b, r_t, s_{t+1})$ in replay buffer $\mathcal{D}$
end for
for each gradient update step **do**
  Sample a minibatch $\{(s, a_c, a_b, r, s')\}$ from $\mathcal{D}$
  Compute target value according to Eq 6
  Update Q-functions by minimizing according to Eq. 7
  Update policy π_θ using Eq. 8
  Update target networks ϕ_i'
end for

IV. Experiments

The setup for our experiments is described in Sec. IV-A. We provide details on the baselines used in Sec. IV-B. We conduct an analysis on our results in Sec. IV-C. We compare the effect of deterministic base policy in Sec. III-C. Finally, we describe our ablation studies in Sec. IV-E.

A. Experiment Setup

We define the robotic manipulation tasks for our experiments in Sec. IV-A1. All environments use sparse rewards and state-based observations with task visualizations in Fig. 1. We run all experiments with 5 seeds and provide hyperparameters used in the supplementary material.

1) Environments: **Robosuite** - We evaluate the performance of the Panda robotic arm on three distinct tasks from the Robosuite simulator. We receive a sparse reward of 1 for successfully completing the task and 0 otherwise. We use the following tasks: 1) **Lift** task, where the robot arm must pick up a block placed at a random initial position on the table. 2) **Can** task, where the robot arm has to pick up a can from one table and place it in the top right corner of another table. 3) **Square** task, where the robot arm must pick up a square nut from the table and place it onto a square bolt.

Franka Kitchen - The *Franka Kitchen* environment from the D4RL benchmark [8] features a Franka robot operating within a kitchen setup that includes a microwave, kettle, overhead light, cabinets, and an oven. The robot is required to interact with these objects to achieve a multitask goal configuration. It receives a reward of 1 for successfully completing each of the 4 sub-goals, and we report the normalized reward for each trajectory. The environment includes three datasets for the *Franka Kitchen* task from the D4RL benchmark: 1) **Kitchen Complete** - This dataset is limited in size and consists solely of positive demonstrations. We perform additional experiments with the other two datasets. 2) **Kitchen Mixed** - This dataset includes undirected demonstrations, with a portion of them successfully solving the task. 3) **Kitchen Partial** - This dataset also contains undirected demonstrations, but none of them fully solve the task. However, each demonstration successfully addresses certain components of the task.

2) Base Policies: We consider two kinds of IL base policies to test the robustness of our algorithm :

GMM-based policy : We utilize Gaussian mixture model-based behavior cloning (BC) policies from [17], which has an RNN backbone. To introduce an additional challenge, we train policies for the *Lift* and *Can* tasks using noisier multi-human demonstrations. Since the *Square* task is inherently challenging, we use proficient-human demonstrations, as even high-quality demonstrations struggle to completely solve the task.

Diffusion-based policy : To ensure a consistent comparison, we use the same Diffusion policies from DPPO [21] for our experiments. For Robosuite tasks, they use noisier multi-human demonstrations, while for the *Franka Kitchen* environments, they directly use datasets from D4RL benchmark.

B. Baselines

We compare our proposed approach against finetuning methods, demo augmented RL methods and other Residual RL methods.

Finetuning methods : As a baseline for Diffusion-based policies, we use the recently proposed **DPPO** [21]. DPPO formulates the denoising process of the Diffusion policy as a separate MDP, effectively modeling the entire trajectory as a sequence of MDPs. The policy is then optimized using policy gradient across this entire chain of MDPs.

Demo augmented RL methods : We compare our method against three demo-augmented RL approaches. **IBRL** [13] maintains both an IL policy trained on demonstrations and an RL policy trained from scratch. During environment interaction and RL training, both policies propose actions, and the action with the highest Q value is selected. A variant of this approach, **IBRL-RPL**, replaces the RL policy learned from scratch with a residual policy. We conduct experiments with both versions of IBRL.

Residual RL methods : The method most closely related to ours is **Policy Decorator** [28], which also aims to mitigate excessive exploration in Residual RL. It addresses this by sampling actions uniformly from both the residual and base policies while gradually decreasing the proportion of actions taken from the base policy. Additionally, it bounds the actions

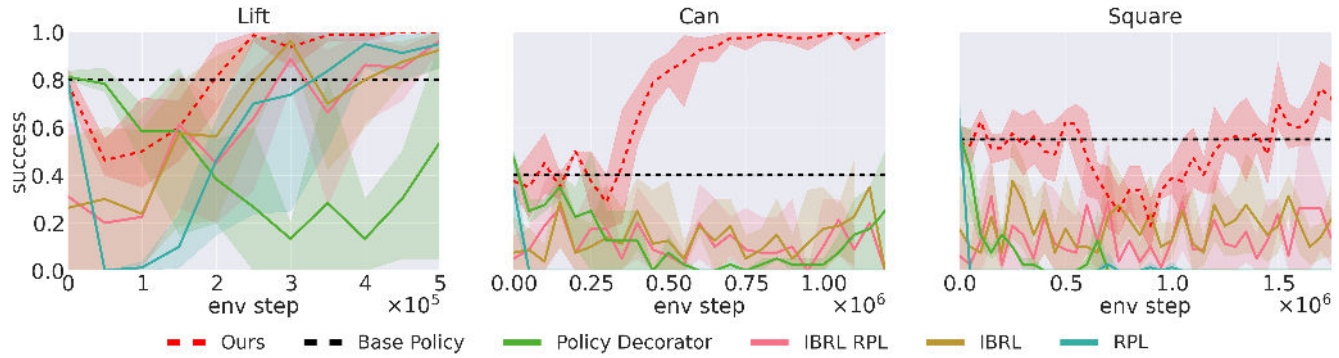

Fig. 3: Results on Robosuite environments with a GMM base policy. Our method is able to outperform all other baselines in all tasks. The error bars indicate 95% confidence interval.

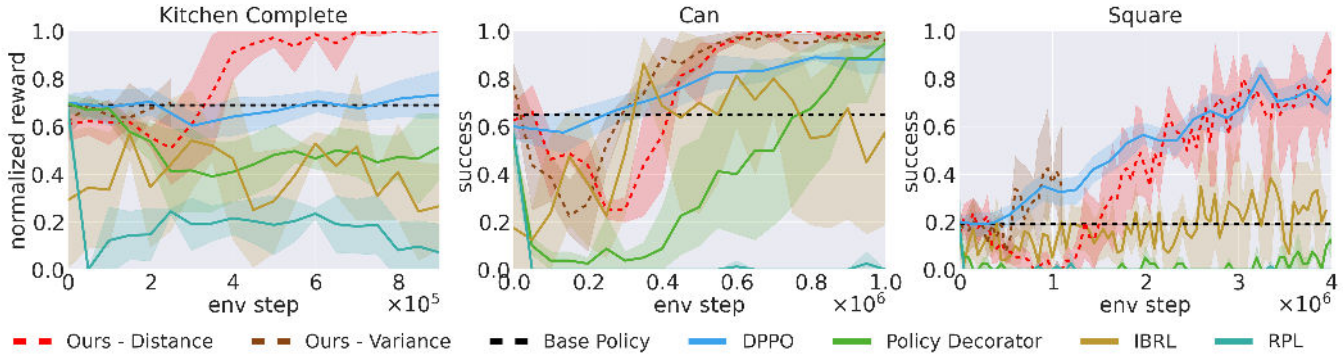

Fig. 4: Results on Franka Kitchen and Robosuite environments with a Diffusion base policy. Our method is able to outperform all baselines for *Kitchen Complete* and *Can* task, and has comparable performance for *Square* Task. The error bars indicate 95% confidence interval.

of the residual policy. For completeness, we also compare our approach with the standard **Residual RL** [26, 14] algorithm, incorporating our proposed modifications to the critic.

C. Results and Analysis

We analyze the results for GMM-based policies in Sec. IV-C1 and the results of Diffusion policies in Sec. IV-C2. We compared the performance of our method with both distance-to-data and ensemble variance as the uncertainty metric for these results and further ablations are done in Sec. VII-E1.

1) GMM-based policies: We present the results of our experiments with GMM based policies in Figure 3. We plot the success rate over the course of environment interactions. Our method (red line) is able to outperform all the baselines in the three Robosuite environments. We note that there is an initial dip in the performance for our method where the residual policy is in exploration phase, but it is stable once the exploration phase ends. IBRL performs the best out of the other baselines with its performance comparable to our method for the *Lift* Task, though the initial dip in performance is more significant, and it is still unstable afterwards. We performed a hyperparameter sweep for the two additional parameters of Policy Decorator, namely the residual bound and the decay rate. Further detail of baselines can be found in appendix.

2) Diffusion policies: We present the results for Diffusion policies in Figure 4. We run the experiments for the *Kitchen Complete* environment of the D4RL benchmark. In Robosuite, we perform experiments in the *Can* and *Square* environments, excluding the *Lift* environment because of the near-optimal performance of Diffusion policy on that task. We performed a hyperparameter sweep of Policy Decorator as mentioned in Sec. IV-C1. Our approach is able to achieve higher success rates than all the baselines in the *Kitchen Complete* environment. In the *Can* environment, although IBRL is able to reach a higher success rate faster than our method, it does not maintain its performance. We note that even though DPPO's performance is stable in *Kitchen Complete* and *Can* environments, its improvement over the initial performance of the base policy is slow. Our method has comparable performance to DPPO for the *Square* task despite an initial dip in the performance. These results suggest that our approach is most promising in scenarios where the initial base policy performance is average, and is comparable in scenarios where the initial base policy performance is bad.

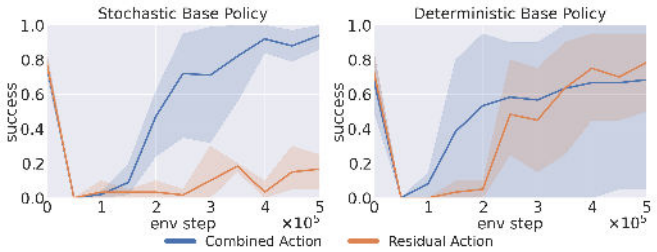

Fig. 5: Learning with combined action and residual action for stochastic and deterministic base policies.

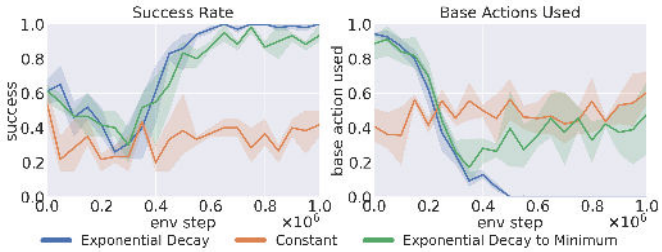

Fig. 6: Ablations for different threshold decay strategies. Decaying exponentially performs the best.

D. Combined Action vs Residual Action

To emphasize the significance of utilizing combined actions for stochastic policies, we compared our modified SAC algorithm for Residual RL without uncertainty estimates to the original Residual RL formulation. For this comparison, we used the GMM policy as the stochastic base policy and a standard MLP policy as the deterministic base policy, applied to the *Lift* task in Robosuite. The results of using combined actions and residual actions during learning for both stochastic and deterministic base policies are shown in Figure 5. The findings reveal that relying solely on the residual action does not yield effective results for stochastic base policies, highlighting the necessity of combining actions in such cases. In contrast, for deterministic base policies, either residual actions or combined actions can be effectively used.

E. Ablations

We ablated our proposed approach with different decaying strategies for the uncertainty threshold in Sec. IV-E1, and with different decay rates in Sec. IV-E2. We present some additional ablations in the supplementary material. All ablations are performed with the Diffusion base policy for the *Can* task.

1) Threshold Decay Strategy: We tried different strategies for decaying the uncertainty threshold τ: exponentially decaying the threshold to zero, exponentially decaying to a minimum threshold, and keeping the threshold constant. We plot the success rate and the percentage of base policy actions used in Figure 6. We observe that exponential decay has the most stable performance out of the three. Exponentially decaying to a minimum threshold converges at a lower success rate compared to exponentially decaying the threshold to 0. The base actions used when decaying the threshold to a minimum value start increasing once that minimum threshold value is reached, signifying that the optimal policy stays within the distribution of base policy. Keeping the threshold value constant restricts residual policy to escape the performance of the initial base policy.

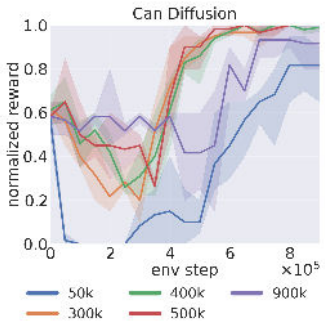

Fig. 7: Ablations for different decay rates.

2) Threshold Decay Rate: We evaluated our algorithm with different decay rates in Figure 7. Lower rates resulted in aggressive exploration, causing performance dips that were hard to recover from, while higher rates slowed the convergence. We need to balance decay rates for effective exploration without hindering convergence. However, our approach is not too sensitive to the chosen decay rate, as the performance is similar for decay rates ranging between the 300k-500k range.

V. Conclusion and Future Work

In this work we propose two improvements to the Residual RL framework to accelerate the learning with stochastic base policies. First, we use uncertainty estimates of the base policy to constrain the exploration of residual learning. Second, we adapt Residual RL to handle stochastic base policies by proposing an asymmetric actor-critic approach in which the critic observes the combined action, while the actor predicts only the residual action. Our method is able to outperform various baselines on tasks from Robosuite and the D4RL Benchmark. While our proposed method demonstrates strong performance, it would also benefit from a more robust epistemic uncertainty metric. We believe that with reliable uncertainty metrics, our approach could also be applied to larger models including robot foundation models.

References

[1] Minttu Alakuijala, Gabriel Dulac-Arnold, Julien Mairal, Jean Ponce, and Cordelia Schmid. Residual reinforcement learning from demonstrations. *arXiv preprint arXiv:2106.08050*, 2021.

[2] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

[3] Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to

refinement–residual rl for precise visual assembly. *arXiv preprint arXiv:2407.16677*, 2024.

[4] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[5] Bernadette Bucher, Karl Schmeckpeper, Nikolai Matni, and Kostas Daniilidis. An adversarial objective for scalable exploration. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2670–2677. IEEE, 2021.

[6] Matthew A Chan, Maria J Molina, and Christopher A Metzler. Hyper-diffusion: Estimating epistemic and aleatoric uncertainty with a single model. *arXiv preprint arXiv:2402.03478*, 2024.

[7] 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*, 2024.

[8] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[9] Georgios Georgakis, Bernadette Bucher, Anton Arapin, Karl Schmeckpeper, Nikolai Matni, and Kostas Daniilidis. Uncertainty-driven planner for exploration and navigation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 11295–11302. IEEE, 2022.

[10] Meghna Gummadi, Cassandra Kent, Karl Schmeckpeper, and Eric Eaton. A metacognitive approach to out-of-distribution detection for segmentation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6642–6649. IEEE, 2024.

[11] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[12] Siddhant Haldar, Jyothish Pari, Anant Rai, and Lerrel Pinto. Teach a robot to fish: Versatile imitation from one minute of demonstrations. *arXiv preprint arXiv:2303.01497*, 2023.

[13] Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning, 2023.

[14] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[15] Sung-Wook Lee and Yen-Ling Kuo. Diff-dagger: Uncertainty estimation with diffusion policy for robotic manipulation. *arXiv preprint arXiv:2410.14868*, 2024.

[16] Yu Tang Liu, Eric Price, Michael J Black, and Aamir Ahmad. Deep residual reinforcement learning based autonomous blimp control. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12566–12573. IEEE, 2022.

[17] 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. *arXiv preprint arXiv:2108.03298*, 2021.

[18] Volodymyr Mnih. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*, 2016.

[19] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[20] Frank Permenter and Chenyang Yuan. Interpreting and improving diffusion models using the euclidean distance function. 2023.

[21] Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.

[22] Jürgen Schmidhuber. *Making the world differentiable: on using self supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments*, volume 126. Inst. für Informatik, 1990.

[23] Gerrit Schoettler, Ashvin Nair, Jianlan Luo, Shikhar Bahl, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5548–5555. IEEE, 2020.

[24] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International conference on machine learning*, pages 8583–8592. PMLR, 2020.

[25] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=agTr-vRQsa.

[26] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[27] HJ Terry Suh, Glen Chou, Hongkai Dai, Lujie Yang, Abhishek Gupta, and Russ Tedrake. Fighting uncertainty with gradients: Offline reinforcement learning via diffusion score matching. In *Conference on Robot Learning*, pages 2878–2904. PMLR, 2023.

[28] Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. *arXiv preprint arXiv:2412.13630*, 2024.

[29] Zhuangzhuang Zhang, Yizhao Wang, Zhinan Zhang, Lihui Wang, Huang Huang, and Qixin Cao. A residual reinforcement learning method for robotic assembly using visual and force information. *Journal of Manufacturing Systems*, 72:245–262, 2024.

VI. Supplementary

A. Environments

The observation space and the action space of each environment used is below -

1) **Lift** - 19 dim observation space with object state and robot end effector state. Action space is 7 DoF end effector pose.
2) **Can** - 23 dim observation space with object state and robot end effector state. Action space is 7 DoF end effector pose.
3) **Square** - 23 dim observation space with object state and robot end effector state. Action space is 7 DoF end effector pose.
4) **Kitchen** - 60 dim observation space with all object states and velocities, and robot joint states and angular velocity. Action space is 9 DoF joint angular velocity and gripper linear velocity.

VII. Base Policies

A. GMM Base Policy

We use Robomimic [17] to train Gaussian Mixture Model based policies with a Recurrent Neural Network backbone.

B. Diffusion policy

We used the same base policies from DPPO [21] to keep the comparisons consistent. The Diffusion policies are trained with an action horizon of 1 and 20 denoising steps.

C. Hyperparameters

We used the same hyperparameters in each environment for both GMM-based and Diffusion-based. We keep same parameters for actor and critic for Robosuite environments and used the advised hyperparametes from the DPPO paper for kitchen environment. Hyperparameter details can be found in Table I. The uncertainty threshold value, U and decay rate values for our proposed approach can be found in Table II.

Environment	Actor & Critic Dimensions	Actor lr	Critic lr
Robosuite	(256,256)	1e-4	1e-4
Kitchen	(256,256,256)	1e-5	1e-3

TABLE I: Hyperparameters used for each environment

Environment	Base Policy	U	Decay Rate
Lift	GMM	1e-6	200k
Can	GMM	2e-5	75k
Square	GMM	5e-5	150k
Kitchen Complete	Diffusion	2.5e-3	200k
Can	Diffusion	4.5e-5	400k
Square	Diffusion	4.5e-5	1M

TABLE II: Uncertainty threshold U and Decay Rate values for our method

Environment	Base Policy	α	H
Lift	GMM	0.1, 0.2, 0.05	400k, 600k
Can	GMM	0.05,0.1,0.2,0.5	400k, 600k, 800k
Square	GMM	0.05, 0.1, 0.5	750k, 1M
Kitchen Complete	Diffusion	0.1, 0.2, 0.3	400k, 600k
Can	Diffusion	0.2, 0.5	400k

TABLE III: Residual bound α and Progressive Exploration schedule H for Policy Decorator

D. Tuning Policy Decorator

Policy Decorator [28] has two hyperparamters namely, α the residual bound which scales the residual action to limit exploration and H to schedule the exploration progressively. According to their paper, the α value is set close to the action scale of demonstration data while it is advised to keep H large as a safe choice. We present the hyperparameters values we used in our sweep in Table III. The authors perform an ablation with DPPO in their appendix but the hyperparmeters were not mentioned and we have reached out to the authors for the same.

E. Ablations

We ablated our proposed approach with more uncertainty metrics in Sec. VII-E1 and with other variations of the kitchen environment in Sec. VII-E2.

1) Uncertainty Metrics: We tested our algorithms with two kinds of uncertainty metrics: 1) **Distance-to-data** as defined in Eq. 2, and 2) **Sample Variance**, where sampling multiple actions from the Diffusion policy for the same observed state is used to calibrate confidence. Low variance in the sampled actions indicates that the policy is confident in its predictions, where high variance indicates higher uncertainty. We sample 5 action predictions to calculate the variance. We can observe in Figure 8 that the Distance-to-data metric worked the best for the *Can* task, where we only have task-specific demonstration data in the training data. This is because Distance-to-data provides only epistemic uncertainty estimates, unlike sample variance of the Diffusion policy which contains both aleatoric and epistemic uncertainty. We further expand on the poor performance on *Kitchen Partial* and *Kitchen Mixed* below.

2) Kitchen Environments: Next, we tested our method on the other two variations of the kitchen environments, *Kitchen Partial* and *Kitchen Mixed*. As seen in Figure 9, our method is not able to outperform the base policy. One key assumption we make in our algorithm is that when the base policy is certain, it will also be correct. However, this assumption does not hold true for these two environments: the base policies for these two environments are trained on the random play data, so the confidence and correctness of the policy is not strongly correlated, resulting in poorer performance. Thus, even with different uncertainty metrics in Sec. VII-E1 the performance remains subpar.

Fig. 8: Ablations for different uncertainty metrics. Distance-to-data has the best performance.

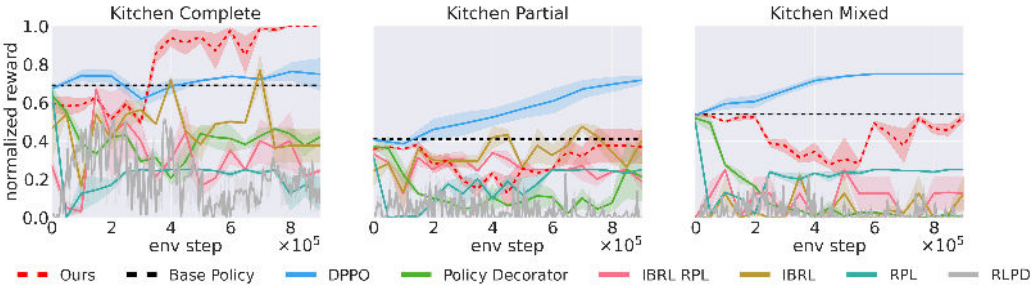

Fig. 9: Ablations for the other kitchen environments that contain undirected demonstrations.