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

Accelerating Residual Reinforcement Learning With Uncertainty Estimation

Lakshita Dodeja, Karl Schmeckpeper, Shivam Vats, Thomas Weng, Mingxi Jia, George Konidaris, and Stefanie Tellex

***Abstract*—Residual Reinforcement Learning (RL) is a popular approach for adapting pretrained policies by learning a lightweight residual policy that provides corrective actions. While Residual RL is more sample-efficient than 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 simulation benchmark environments. We also deploy our learned policies in the real world to demonstrate their robustness with zero-shot sim-to-real transfer.**

***Index Terms*—Reinforcement learning (RL), deep learning methods, machine learning for robot control.**

I. Introduction

RESIDUAL Reinforcement Learning improves the performance of pretrained policies by training a separate policy to output corrective actions [1], [2]. Directly fine-tuning the pretrained policy is often computationally expensive, especially in the case of policies with a large number of parameters [3], [4], and is prone to instability [5]. 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 however the base policy was built.

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 [6], [7]. Furthermore, recent advancements in imitation learning leverage highly effective stochastic policies: Gaussian Mixture Model-based policies [8] and Diffusion policies [3] excel at modeling complex, multi-modal distributions. In such cases, residual RL algorithms that assume a deterministic base policy [1], [2] are not suitable.

We address these limitations by proposing two improvements to Residual RL that enhance its sample efficiency and make it more 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 function only for the residual action a_r, i.e. $Q(s, a_r)$, implicitly assuming that the base policy's action can be inferred from the state s. This is insufficient when dealing with stochastic base policies, since they sample different actions given the same state. In the stochastic setting, the Residual RL agent cannot infer a single base action, making it difficult to learn a good residual action. Prior works have attempted to solve this by using learned bottleneck features of the base policy as a prior for residual learning [9] or by augmenting the observed state with the base action for on-policy learning [10]. We propose an asymmetric actor-critic 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 critic invariant to the split between the residual and base action.

We evaluate our approach on a variety of manipulation tasks from Robosuite[8] and Franka Kitchen tasks from D4RL[11].

Received 24 June 2025; accepted 22 October 2025. Date of publication 24 November 2025; date of current version 3 December 2025. This article was recommended for publication by Associate Editor H. Kasaei and Editor J. Kober upon evaluation of the reviewers' comments. The work Lakshita Dodeja was supported by the NASA-Kennedy Award Efficient and High Quality Space Robotic VR Teleoperation through Neural Scene and Object Reconstruction under Award Number 80NSSC23M0075. The work of Shivam Vats and George Konidaris was supported in part by the Office of Naval Research (ONR) under Grant REPRISM MURI N000142412603 and ONR Grant N00014-22-1-2592, in part by the National Science Foundation (NSF) under Grant 1955361, and in part by Robotics and AI Institute. *(Corresponding author: Lakshita Dodejaemail.)*

Lakshita Dodeja, Shivam Vats, Mingxi Jia, and George Konidaris are with Brown University, Providence, RI 02912 USA (e-mail: lakshita_dodeja@brown.edu).

Karl Schmeckpeper and Thomas Weng are with Robotics and AI Institute, Cambridge, MA 02142 USA.

Stefanie Tellex is with Brown University, Providence, RI 02912 USA, and also with Robotics and AI Institute, Cambridge, MA 02142 USA.

Paper homepage : lakshitadodeja.github.io/uncertainty-aware-residual-rl/.

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3636808, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3636808

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

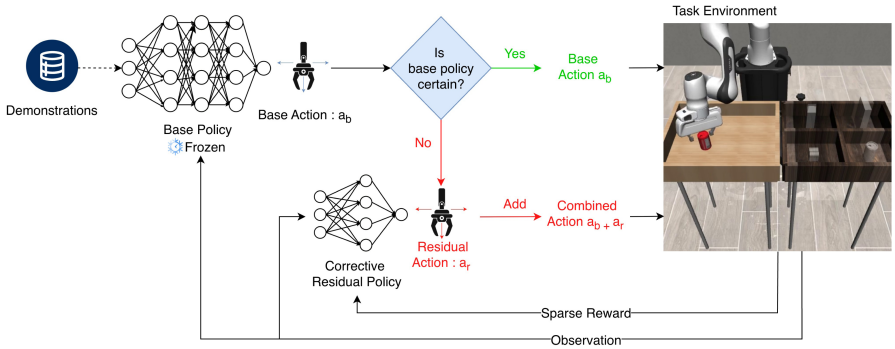

Fig. 1. We propose two improvements to accelerate Residual RL : 1) We use uncertainty estimation to constrain exploration around the base policy. 2) We modify the off-policy critic to learn the Q function for the combined action. We test our method with two uncertainty metrics, i.e. distance to data and ensemble variance.

We test our approach on both Gaussian Mixture Model and Diffusion base policies. Finally, we compare our approach with a state-of-the-art finetuning method[12], a demo-augmented RL method[13], and other Residual RL methods[1], [2], [14]. Our proposed approach outperforms or is comparable to other baselines in all tasks. We also perform several ablation studies to test various aspects of our algorithm and deploy the learned policies on a real robot to demonstrate sim-to-real transfer. Our proposed approach is visualized in Fig. 1 with the main contributions summarized as follows:

1) We present a novel algorithm to accelerate Residual Reinforcement Learning using uncertainty estimates.
2) We modify off-policy Residual Reinforcement Learning to work with stochastic base policies using an asymmetric actor-critic approach.
3) We validate our method on robotic manipulation tasks from different simulators and against several baselines. We demonstrate zero-shot sim-to-real transfer of our learned policies.

II. Background

In Residual RL, we assume that we have a suboptimal base policy π_b. The objective is to learn a lightweight residual policy π_r, on top of the base policy that gives a corrective action a_r, producing a more accurate and robust combined policy. Thus, the final policy executed in the environment is:

$$\pi = \pi_b(s) + \pi_r(s) \tag{1}$$

Residual RL transforms the original Markov decision process (MDP) formulation $M =< S, A, R, T, \gamma >$ to the residual MDP (RMDP) formulation $M_r =< S, A_r, R, T_r, \gamma >$ [1], where 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 the original transition function as follows:

$$T_r(s, a_r, s') = T(s, \pi_b(s) + a_r, s'). \tag{2}$$

III. Related Work

Residual RL for Stochastic Base Policies: Residual RL, first introduced for robotics in [2] and [1], 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[3] and GMM-based policies[8] are non-deterministic, making the original Residual RL formulation insufficient due to the lack of information about the base policy. Some works introduce noise in the base action to enhance robustness and induce stochasticity[6], [7]. Other works inform residual learning about the base policy by conditioning it on the learned bottleneck features of the base policy [9], and incorporating the base action in the observed state to inform the residual policy for on-policy learning [10]. In contrast, our work modifies off-policy RL to learn the Q function for the combined action taken in the environment (i.e. the sum of base and residual action), and the actor 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[14] also uses the combined action as an input to their critic for Residual RL, but we do a thorough quantitative evaluation to show that it significantly improves performance.

Imitation Learning and Residual RL: Several works have explored the integration of Imitation Learning (IL) base policies with Residual RL. Residual RL has been applied to insertion tasks [15], where demonstrations are incorporated as an auxiliary behavior cloning (BC) loss during RL training [16]. FISH [17] employs a non-parametric base policy alongside a residual policy that uses optimal transport matching against offline demonstrations as the reward. BeT [18] 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 using IL actions as alternative proposals for both online exploration and critic updates. Closest to our method is Policy Decorator [14], which learns bounded residual actions

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

using controlled exploration. Unlike Policy Decorator, which uniformly samples actions from the base and residual policies, we use uncertainty estimates of the base policy to decide when to learn and apply corrective residual actions.

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. Various approaches have been proposed to quantify uncertainty, including distance-based techniques[19], [20], ensemble-based techniques [21], [22], [23], and learning another model to estimate uncertainty [24], [25], [26]. SGP[20] 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. Diff-DAgger[27] uses the loss function of a Diffusion model to estimate uncertainty, where a higher loss indicates greater uncertainty. Our algorithm is agnostic to the uncertainty quantification method, and we test our approach with different uncertainty metrics.

IV. Uncertainty Aware Residual RL for Stochastic Policies

We describe how to incorporate uncertainty estimates in Residual RL in Sec. IV-A and our modified off-policy RL algorithm in Sec. IV-B.

A. 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 base policy actiona_b in the environment; we only use a corrective residual action a_r when the base policy is uncertain. Our proposed approach is agnostic to the uncertainty quantification method, and we demonstrate our method with two distinct uncertainty metrics: 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[20]. For a dataset D where each state has F features, we estimate uncertainty using the minimum L2 norm of the current state s to all states $d \in D$:

$$\text{uncertainty}_d(s) = \min_{d \in D} \sqrt{\sum_{i=1}^{F} (d_i - s_i)^2}. \tag{3}$$

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

$$\text{uncertainty}_e(s) = \frac{1}{N} \sum_{b=1}^{N} \left(\pi_b(a \mid s) - \frac{1}{N} \sum_{i=1}^{N} \pi_i(a \mid s) \right)^2. \tag{4}$$

We can use our desired uncertainty metric with a threshold τ to measure the confidence of the base policy. This can be formulated as

$$a_{\text{taken}} = \begin{cases} a_{\text{b}} & \text{if uncertainty} < \tau \\ a_{\text{b}} + a_{\text{r}} & \text{otherwise.} \end{cases} \tag{5}$$

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

$$\tau = U * e^{\frac{-\text{step}}{\text{decay rate}}}. \tag{6}$$

The uncertainty threshold τ ultimately decays to 0 to let the residual policy take over. We perform ablations for different decaying strategies in Sec. V-G1.

B. Optimizing Residual RL for Stochastic Policies

The original Residual RL algorithms are formulated to learn only using 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:

$$Q(s, a_r) = \mathbb{E}_{s' \sim P} \left[R(s, a_r, s') + \gamma V^{\pi}(s') \right]. \tag{7}$$

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. 2 to retreat back to the original MDP formulation. Consequently, the Q function is learned for the combined action a_c, which is the actual action used during environment interaction :

$$Q(s, a_c) = \mathbb{E}_{s' \sim P} \left[R(s, a_b + a_r, s') + \gamma V^{\pi}(s') \right] \tag{8}$$

Previously, ResiP[10] 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 for off-policy RL, providing the necessary information about the base policy to the Q function while also making it invariant to the split between the residual and base actions. Specifically, we modify Soft Actor-Critic [28] in the following ways (changes marked in green). Initially, we store both the base action a_b and the combined action a_c in the replay buffer. While computing the target values, we add the base action a_b to the residual action a_r sampled from the actor:

$$\begin{aligned} y(r, s', d) = r + \gamma(1 - d) * \\ \left[\min_{i=1,2} Q_{\phi'_i}(s', a'_r + a'_b) - \alpha \log \pi_r(a'_r | s') \right], \\ a'_r \sim \pi_r(\cdot | s'). \end{aligned} \tag{9}$$

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

Fig. 2. We test our proposed approach on the *Lift*, *Can*, and *Square* tasks from Robosuite[8] and the *Franka Kitchen* Task from D4RL[11].

Algorithm 1: Uncertainty aware Residual RL.

1: Initialize parameters of residual policy π_r, Q-functions Q_{ϕ_1}, Q_{ϕ_2}, and temperature α
2: Initialize target Q-function parameters $\phi_1' \leftarrow \phi_1$, $\phi_2' \leftarrow \phi_2$
3: Initialize base policy π_b
4: **for** each environment step **do**
5: Sample residual action from actor
6: $a_r \sim \pi_r(s_t)$
7: Sample base action from base policy
8: $a_b \sim \pi_b(s_t)$
9: Calculate uncertainty threshold τ for the base policy, Eq. 6
10: Calculate the uncertainty in base policy, Eq. 3
11: Select the action to be taken in the environment, Eq. 5
12: Observe next state s_{t+1} and reward r_t
13: Store $(s_t, a_c, a_b, r_t, s_{t+1})$ in replay buffer $\mathcal{D}$
14: **end for**
15: **for** each gradient update step **do**
16: Sample a minibatch $\{(s, a_c, a_b, r, s')\}$ from $\mathcal{D}$
17: Compute target value according to Eq 9
18: Update Q-functions by minimizing according to Eq. 10
19: Update policy π_θ using Eq. 11
20: Update target networks ϕ_i'
21: **end for**

When updating the Q function, we use the combined action stored in the replay buffer:

$$J_Q(\phi_i) = \mathbb{E}\left[\left(Q_{\phi_i}(s, a_c) - y(r, s', d)\right)^2\right], \quad i = 1, 2. \quad (10)$$

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

$$J_\pi(\theta) = \mathbb{E}\left[Q_{\phi_i}(s, a_r + a_b) - \alpha \log \pi_r(a_r|s)\right], \quad i = 1, 2, \quad a_r \sim \pi_r(\cdot|s). \quad (11)$$

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

V. Experiments

In this section we describe our experiment setup, baselines, results and ablations.

A. Experiment Setup

1) Environments: We conduct evaluations on the environments visualized in Fig. 2 and described below. All environments use sparse rewards and state-based observations.

Robosuite: We evaluate on three tasks from the Robosuite simulator [8]: 1) **Lift**, where the robot arm must pick up a block placed at a random initial position on the table. 2) **Can**, 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**, 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[11] features a Franka robot that is required to interact with various 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.

Gaussian Mixture Model-based policy: We utilize GMM-based behavior cloning (BC) policies from[8], 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: We also test our method with a diffusion base policy[3]. To ensure a consistent comparison, we use the same Diffusion policies from DPPO[12] for our experiments. For Robosuite tasks, they use noisier multi-human demonstrations, while for the *Franka Kitchen* environments, they directly use datasets from D4RL.

B. Baselines

Finetuning methods: As a baseline for Diffusion-based policies, we use the recently proposed **DPPO**[12]. 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 gradients across this entire chain of MDPs.

Demo augmented RL methods: We compare our method against two 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.

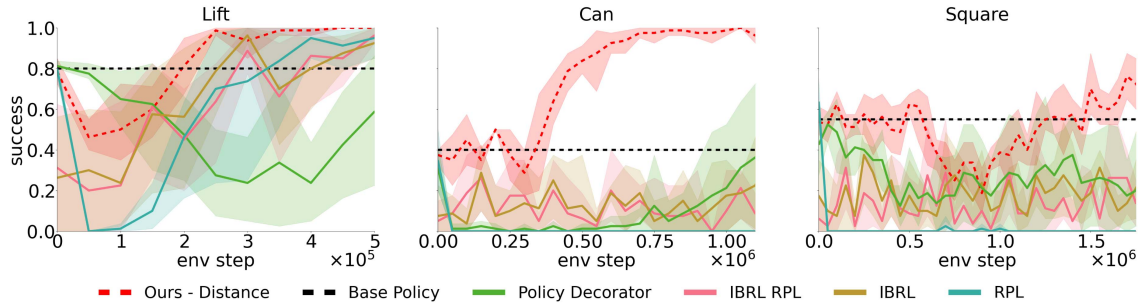

Fig. 3. Results on Robosuite environments with a GMM base policy. Our method is able to outperform all other baselines in all tasks. The error bars indicate 95% confidence interval.

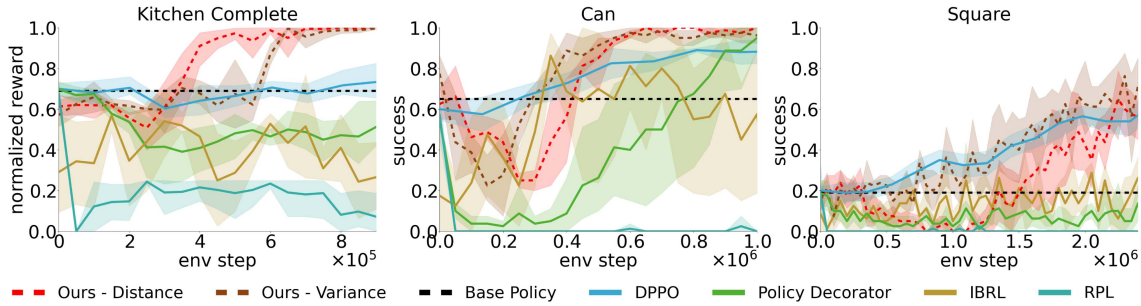

Fig. 4. Results on Franka Kitchen and Robosuite environments with a Diffusion base policy. Our method is able to outperform all baselines for *Kitchen Complete* and *Can* task, and has comparable performance for *Square* Task. The error bars indicate 95% confidence interval.

Residual RL methods: We test our method against the most closely related approach and current state-of-the-art residual learning algorithm, **Policy Decorator**[14], which also aims to mitigate excessive exploration in Residual RL. It addresses this by sampling actions uniformly from both the residual and base policies with a progressive exploration schedule. Additionally, it bounds the actions of the residual policy. For completeness, we also compare our approach with the standard **Residual RL**[1], [2] algorithm, incorporating our proposed modifications to the critic without using uncertainty estimation.

C. Results and Analysis

We evaluate all experiments with 5 seeds over 20 runs and provide hyperparameters in Appendix.

1) GMM-Based Policies: We present the results of our experiments with GMM-based policies in Fig. 3. We plot the success rate over the course of environment interactions. Our method with distance to data metric (red line) outperforms 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 the exploration phase, but it starts improving once the exploration phase ends. IBRL performs the best out of the other baselines, reaching comparable performance 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 with more details in the Appendix D. Our method converges in the same number of timesteps, while Policy Decorator's performance is still improving. We suspect it is due to the more targeted exploration of our method using uncertainty estimates. The standard Residual Policy Learning method only improves over the base policy performance for the Lift Task.

2) Diffusion Policies: We present the results for Diffusion policies in Fig. 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 similar hyperparameter sweep of Policy Decorator as mentioned in Sec. V-C1 with details in Appendix D. Similarly, our method is able to outperform Policy Decorator in all environments. Our approach is able to achieve higher success rates than all the baselines in the *Kitchen Complete* environment with both types of uncertainty metrics. 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 compared to our approach, despite an initial dip in the performance. Our method with the ensemble variance metric has comparable performance to DPPO for the *Square* task. These results suggest that our approach is most promising when the initial base policy performance is average, and is comparable in scenarios where the initial base policy performance is bad. Also, we note that distance-to-data as a metric works better than ensemble variance for the *Kitchen Complete* environment but not the *Can* and *Square* environment. We hypothesize that this is because of the high quality demonstrations in the *Kitchen Complete* environment, whereas the noisier multi-human demonstrations of *Can* and *Square* tasks result in a noisier distance-to-data metric. Nevertheless, our method converges for both the metrics.

D. Combined Action Vs. Residual Action

To emphasize the significance of utilizing combined actions for stochastic policies, we compared our modified SAC

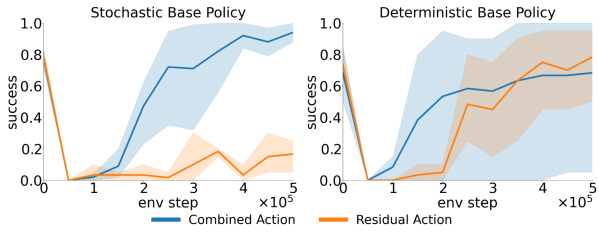

Fig. 5. Learning with either the residual or the complete action works well with deterministic base policies, but learning with complete action is required for stochastic base policies.

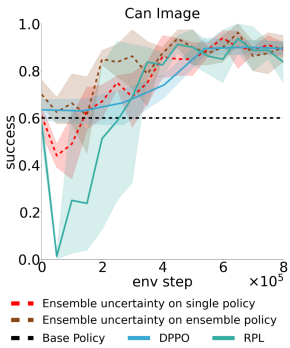

Fig. 6. Results on the Can-Image Task.

algorithm for Residual RL *without* uncertainty estimates against 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 for both stochastic and deterministic base policies are shown in Fig. 5. The findings reveal that relying solely on the residual action does not yield effective results for stochastic base policies, highlighting the necessity of using our combined action formulation in such cases. In contrast, for deterministic base policies, either residual actions or combined actions can be used.

E. Image Based Experiments

We evaluated our algorithm on the *Can* task with image observations. Given that distance-to-data is a less reliable metric in high dimensional input spaces such as images, we instead used ensemble variance to quantify uncertainty. We explored two strategies: first, using ensemble variance to enhance the performance of a single policy, and second, leveraging it to improve an ensemble of policies. Our method was compared against DPPO, the strongest baseline for finetuning diffusion policies, and Residual Policy Learning (RPL) without uncertainty estimation. The results of our algorithm on the image-based *Can* Task are shown in Fig 6. The results indicate that the ensemble-based approach starts with strong performance and avoids early dips, likely due to the robustness offered by ensembling. In contrast, the naive RPL strategy initially crashes to 0% performance after the initial online learning steps, though it recovers over time. DPPO, as observed previously, maintains

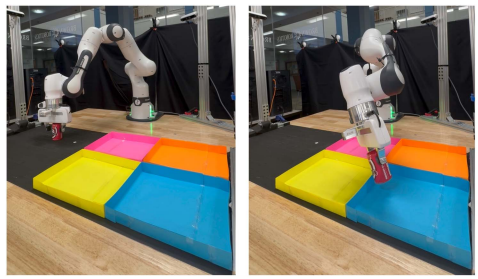

Fig. 7. Sim-to-Real setup for the Robomimic Can task.

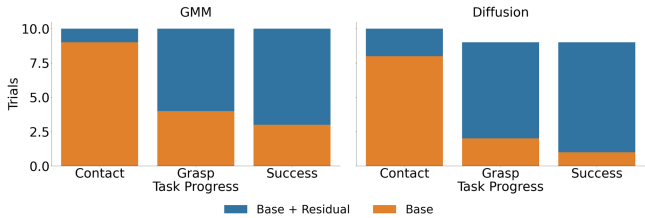

Fig. 8. Results for the policy deployment in real world.

steady but slow improvement. These findings show our approach also performs well in image-based environments.

F. Real World Experiments

We deployed our learned policy for the *Can* task in the real world in a zero-shot setup as shown in Fig 7. The task is set up similarly to the simulated version, where the robot has to pick up the can from one side of the table and place it in the blue bin on the other side of the table. We used Grounded SAM[29] and Foundation Pose[30] to get the real-time state of the can object. We evaluated four policies in our real-world setup: 1) GMM base, 2) GMM base + Residual, 3) Diffusion base, and 4) Diffusion base + Residual. We compare the performance in terms of task progression divided into three stages: contacting, grasping, and placing. Fig 8 shows the performance of each policy across 10 trials. Policies learned with Residual RL retain nearly all of their original performance in simulation without any domain randomization, whereas base policies struggled. This supports the observation that policies trained using RL tend to be more robust than behavior cloning policies, as they benefit from richer interaction with the environment.

G. Ablations

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 success rate and the percentage of base policy actions used in Fig. 9. 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

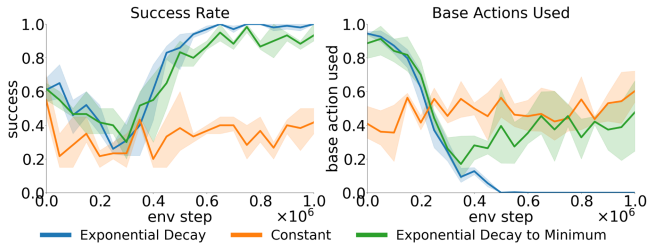

Fig. 9. Ablations for different threshold decay strategies. Decaying exponentially performs best.

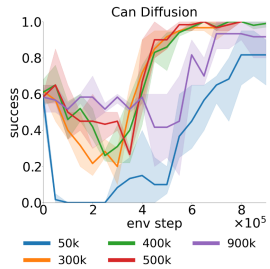

Fig. 10. Higher decay rates result in slower convergence whereas lower decay rates results in aggressive exploration.

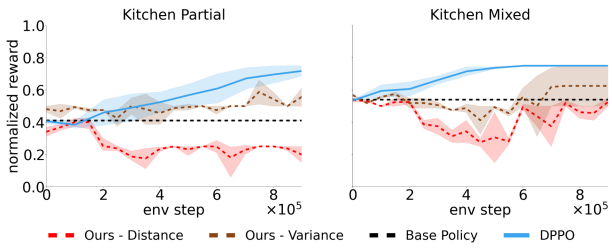

Fig. 11. Ensemble variance is a better uncertainty metric for Kitchen Partial and Kitchen Mixed environments as the training data also contains random trajectories.

increasing once that minimum threshold value is reached, signifying that the optimal policy stays within the distribution of the base policy. Keeping the threshold value constant restricts the residual policy from escaping the performance of the initial base policy.

2) Threshold Decay Rate: We evaluated our algorithm with different decay rates in Fig. 10. 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 robust to the chosen decay rate, as the performance is similar for decay rates ranging between 300 k and 500 k.

3) Kitchen Environments: We tested our method on the other two variations of the kitchen environments, *Kitchen Partial* and *Kitchen Mixed*. As seen in Fig. 11, our method with the distance to data metric 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. 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 are not strongly correlated, resulting in poorer performance. However, ensemble variance is able to provide a better uncertainty metric when random play data is involved and results in improved performance.

VI. Conclusion and Future 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 exploration in 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. 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. In the future, we would also like to dynamically set the values of uncertainty threshold decay based on the performance of the base policy.

Appendix

A. Environments

The observation space of each environment used is below:

1) **Lift** - 19 dim obs space with object state and robot eef.
2) **Can** - 23 dim obs space with object state and robot eef.
3) **Square** - 23 dim obs space with object state and robot eef.
4) **Kitchen** - 60 dim obs space with all object states and velocities, and robot joint states and angular velocity.

Action space for all Robosuite environments is 7 DoF end effector pose while for Kitchen environment is the 9 DoF joint angular velocity and gripper linear velocity.

B. Base Policies

GMM Base Policy - We use Robomimic[8] to train Gaussian Mixture Model based policies with a Recurrent Neural Network backbone.

Diffusion policy - We used the same base policies from DPPO[12] 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 the same parameters for actor and critic for Robosuite environments and use the advised hyperparameters from the DPPO paper for the kitchen environment. Hyperparameter details can be found in Table I. The uncertainty threshold value U and decay rate values for our proposed approach with distance-to-data and ensemble variance metric can be found in Table II.

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

TABLE I
HYPERPARAMETERS USED FOR EACH ENVIRONMENT

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

TABLE II
UNCERTAINTY THRESHOLD U AND DECAY RATE VALUES FOR DISTANCE-TO-DATA AND ENSEMBLE VARIANCE

Environment	Base Policy	U	Decay Rate
Distance-to-data			
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
Ensemble variance			
Kitchen Complete	Diffusion	0.5	200k
Can	Diffusion	0.2	500k
Square	Diffusion	0.4	750k

D. Tuning Policy Decorator

Policy Decorator [14] has two hyperparameters 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 hyperparameter values we used in our sweep in Table III. The authors perform an ablation with DPPO for the square task in their appendix, and we received the hyperparameters from the authors for the same. After looking at their implementation, we observed that they train their RL policies with an expanded action space with an action horizon while we implement the RL policies in the original form with a single action.

TABLE III
RESIDUAL BOUND α AND PROGRESSIVE EXPLORATION SCHEDULE H FOR POLICY DECORATOR

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
Square	Diffusion	0.05	500k

ACKNOWLEDGMENT

The authors would like to thank Ondrej Biza and Ivy He for helpful discussions.

REFERENCES

[1] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual policy learning," *Robot.: Sci. Syst.*, 2015.
[2] T. Johannink et al., "Residual reinforcement learning for robot control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 6023–6029.
[3] C. Chi et al., "Diffusion policy: Visuomotor policy learning via action diffusion," *Int. J. Robot. Res.*, vol. 44, no. 10–11, pp. 1684–1704, 2024.
[4] K. Black et al., "π_0: A vision-language-action flow model for general robot control," *Robot.: Sci. Syst.*, 2025.
[5] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[6] Y. T. Liu, E. Price, M. J. Black, and A. Ahmad, "Deep residual reinforcement learning based autonomous blimp control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 12566–12573.
[7] Z. Zhang, Y. Wang, Z. Zhang, L. Wang, H. Huang, and Q. Cao, "A residual reinforcement learning method for robotic assembly using visual and force information," *J. Manuf. Syst.*, vol. 72, pp. 245–262, 2024.
[8] A. Mandlekar et al., "What matters in learning from offline human demonstrations for robot manipulation," in *Proc. Conf. Robot Learn.*, 2022, pp. 1678–1690.
[9] M. Alakuijala, G. Dulac-Arnold, J. Mairal, J. Ponce, and C. Schmid, "Residual reinforcement learning from demonstrations," 2021, *arXiv:2106.08050*.
[10] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal, "From imitation to refinement–residual RL for precise visual assembly," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2025, pp. 1–8.
[11] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4RL: Datasets for deep data-driven reinforcement learning," 2020, *arXiv:2004.07219*.
[12] A. Z. Ren et al., "Diffusion policy policy optimization," in *Proc. 13th Int. Conf. Learn. Representations*, 2025.
[13] H. Hu, S. Mirchandani, and D. Sadigh, "Imitation bootstrapped reinforcement learning," *Robot.: Sci. Syst.*, 2024.
[14] X. Yuan, T. Mu, S. Tao, Y. Fang, M. Zhang, and H. Su, "Policy decorator: Model-agnostic online refinement for large policy model," in *Proc. 13th Int. Conf. Learn. Representations*, 2025.
[15] G. Schoettler et al., "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5548–5555.
[16] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6292–6299.
[17] S. Haldar, J. Pari, A. Rai, and L. Pinto, "Teach a robot to fish: Versatile imitation from one minute of demonstrations," in *Proc. Robot.: Sci. Syst.*, 2023.
[18] N. M. M. Shafiullah, Z. J. Cui, A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," in *Proc. 36th Conf. Neural Inf. Process. Syst.*, 2022, pp. 22955–22968. [Online]. Available: https://openreview.net/forum?id=agTr-vRQsa
[19] F. Permenter and C. Yuan, "Interpreting and improving diffusion models from an optimization perspective," in *Proc. Int. Conf. Mach. Learn.*, 2024, pp. 40461–40483.
[20] H. T. Suh, G. Chou, H. Dai, L. Yang, A. Gupta, and R. Tedrake, "Fighting uncertainty with gradients: Offline reinforcement learning via diffusion score matching," in *Proc. Conf. Robot Learn.*, 2023, pp. 2878–2904.
[21] G. An, S. Moon, J.-H. Kim, and H. O. Song, "Uncertainty-based offline reinforcement learning with diversified Q-ensemble," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 7436–7447.
[22] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak, "Planning to explore via self-supervised world models," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 8583–8592.
[23] G. Georgakis, B. Bucher, A. Arapin, K. Schmeckpeper, N. Matni, and K. Daniilidis, "Uncertainty-driven planner for exploration and navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 11295–11302.
[24] M. Chan, M. Molina, and C. Metzler, "Estimating epistemic and aleatoric uncertainty with a single model," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, vol. 37, pp. 109845–109870.
[25] B. Bucher, K. Schmeckpeper, N. Matni, and K. Daniilidis, "An adversarial objective for scalable exploration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 2670–2677.
[26] M. Gummadi, C. Kent, K. Schmeckpeper, and E. Eaton, "A metacognitive approach to out-of-distribution detection for segmentation," in *Proc. 2024 IEEE Int. Conf. Robot. Automat.*, 2024, pp. 6642–6649.
[27] S.-W. Lee and Y.-L. Kuo, "Diff-Dagger: Uncertainty estimation with diffusion policy for robotic manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*2025, pp. 4845–4852, doi: 10.1109/ICRA55743.2025.11127730.
[28] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[29] T. Ren et al., "Grounded SAM: Assembling open-world models for diverse visual tasks," 2024, *arXiv:2401.14159*.
[30] B. Wen, W. Yang, J. Kautz, and S. Birchfield, "FoundationPose: Unified 6D pose estimation and tracking of novel objects," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 17868–17879.