Grasping with Chopsticks: Combating Covariate Shift in Model-free Imitation Learning for Fine Manipulation

Liyiming Ke, Jingqiang Wang, Tapomayukh Bhattacharjee, Byron Boots and Siddhartha Srinivasa

***Abstract*— Billions of people use chopsticks, a simple yet versatile tool, for fine manipulation of everyday objects. The small, curved, and slippery tips of chopsticks pose a challenge for picking up small objects, making them a suitably complex test case. This paper leverages human demonstrations to develop an autonomous chopsticks-equipped robotic manipulator. Due to the lack of accurate models for fine manipulation, we explore model-free imitation learning, which traditionally suffers from the *covariate shift* phenomenon that causes poor generalization. We propose two approaches to reduce covariate shift, neither of which requires access to an interactive expert or a model, unlike previous approaches. First, we alleviate single-step prediction errors by applying an invariant operator to increase the data support at critical steps for grasping. Second, we generate synthetic corrective labels by adding bounded noise and combining parametric and non-parametric methods to prevent error accumulation. We demonstrate our methods on a real chopstick-equipped robot that we built, and observe the agent's success rate increase from 37.3% to 80%, which is comparable to the human expert performance of 82.6%.**

I. Introduction

Fine manipulation—cutting your fingernails, inserting a straw into a cup, locking a necklace clasp—are common in everyday tasks. Although complex end effectors are inherently suited to fine manipulation by tailoring their design to the problem [1], simple tools are easier to build, deploy, and are ubiquitous in industrial manipulators. We study robotic fine manipulation using simple tools. However, the practicality of simple tools come at the cost of complexity to control [2]. The one-size-fit-all design of simple tools shifts the burden to control policies. E.g. picking noodles up with a fork using a twirling motion requires sophisticated control policies and the same task would be almost impossible with a spoon. We have yet to determine which simple tools are capable of solving complex fine manipulation tasks.

We turn to a fine manipulation tool that humans have demonstrated incredible dexterity with: chopsticks. Their small, curved, and slippery tips require precise movements for grasping small and rigid objects such as a toy marble. Their limited allowance for failures makes them a suitably complex test case for evaluating fine manipulation tasks. With sophisticated control policies, humans have used chopsticks to pick up food items with varying physical characteristics including size, shape, deformability. The efficacy of chopsticks' design has inspired researchers to adapt them for diverse robotic applications, such as surgery [3]–[5], micromanipulation [6], and meal assistance [7], [8]. Noticeably, humans have demonstrated impressive adaptability in teleoperating a robot equipped with chopsticks to pick up hard-to-grasp small objects [9]. As chopsticks are ubiquitous, familiar to humans, and easy to track, they provide an additional advantage that allows us to collect human data that we can leverage for data-driven algorithms. Therefore, we choose chopsticks, a simple tool familiar to billions of humans, as an example to learn and automate fine manipulation strategies from human demonstrations [10], [11].

University of Washington, Seattle WA 98105 USA. {kayke, jwq123, tapo, bboots, siddh}@uw.edu

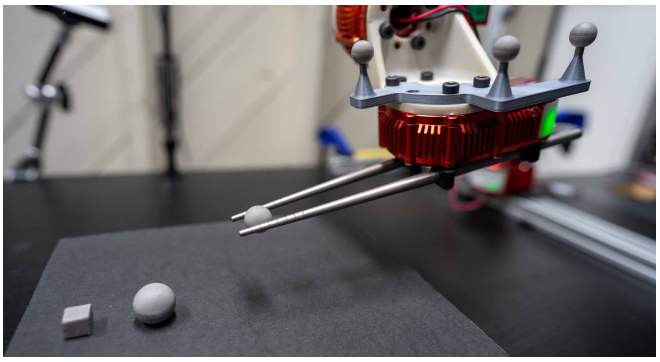


Fig. 1: Fine manipulation using chopsticks.

To derive a policy from demonstrations, we study imitation learning [12]–[15]. Due to repeatability, actuator backlash and unmodeled dynamics like friction or temperature, it can take tremendous efforts to create simulator models to achieve the level of precision required in fine manipulation tasks [16], [17]. The lack of accurate models motivates our study of *model-free* imitation learning: we have access to demonstration data but not to the expert's policy function or the environment's transition model. Under these conditions, supervised learning methods like *behavior cloning* [18] learn a policy function by matching the expert's action distribution. Minimizing action distribution divergence, however, does not necessarily guarantee the recovery of parsimonious states that lead to task success [19]. A learned agent can suffer from *covariate shift* [20], i.e., compounding errors in the action space that lead the agent to unseen states during test time. This problem can be especially detrimental for fine manipulation, the success of which critically depends on a few steps that usually occur near the end of a trajectory.

To remedy covariate shift, researchers have proposed *interactive* imitation learning methods, such as DAgger [21] and DART [22], to query an expert *online* for corrective labels. DAgger rolls out a learned agent and asks the expert for labels on learner visited states, which can be computationally expensive and unnatural on a teleoperation interface [22]. DART injects noise during data collection, disturbs expert teleoperation, and forces the expert to provide corrective

labels. However, injecting noise during data collection can burden the expert: adding a small amount of random noise for our fine manipulation task, as DART suggests, would require the expert to spend 43% more time on collecting data.[1]

These challenges prompt us to address covariate-shift in model-free imitation learning in a *non-interactive* setting, where we have access to demonstration data but not to an interactive expert. Since covariate shift results from the interplay of single-step errors and their accumulation over time, our key ideas are to (1) increase data support to address single-step errors, and (2) provide corrective labels to address the accumulation of errors. Specifically, we provide:

- *Enhanced data support* by transforming the data to an object-centric frame that preserves the relative transformation between the end effector and object, while making training data denser around the *critical* region for grasp success.
- *Corrective labels by injecting noise* into the collected state, assuming the same action may serve as the *corrective label* for the deviated state. Thus, we implicitly enforce smoothness to the learned policy and tell the agent how to recover from deviated states.
- *Corrective labels by choosing a combination of parametric and non-parametric methods* that improve matching of the action distribution at unseen states. Because of our problem structure, a better match in action distribution leads to a higher likelihood of matching the state distribution, preventing error accumulation.

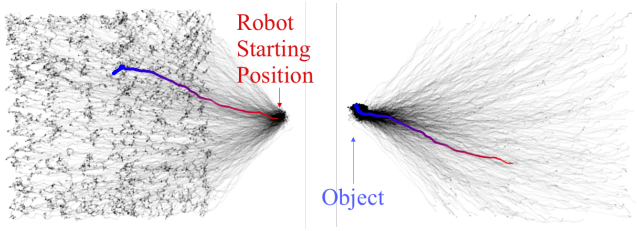
We demonstrate our proposal's effectiveness on a physical robot equipped with chopsticks to pick up small cube- and ball-shaped objects, as shown in Fig. 1. Our proposed agent achieves 60% success rates picking up even the most challenging item, a small ball, whereas a naive behavior cloning agent has only a 12% success rate. Our agent achieves an 80% average success rate picking up all three objects tested, comparable to the expert human performance of 82%. We conduct ablation tests, visualize the resulting states' distribution, and observe a smaller covariate shift from our proposed agents. We also validate the generality of the noise injection method on several Mujoco simulated tasks.

Our promising empirical results, based on pragmatic assumptions of data support and policy smoothness, open the door for further theoretical analysis of combating covariate shift. Furthermore, although we have focused on the *non-interactive* setting, our techniques directly transfer to the *interactive* setting, enhancing robustness while reducing user burden.

II. Methods

A. Transform: Increasing Data Support

Our goal is to develop an agent that can generalize from demonstration data to predict an action for any query state. However, we lack data support for some states (e.g., the "unseen state" during rollout). We propose to apply an invariant operator to transform the data, making it denser around the region of interest and thus increasing the data support.



(a) Robot-centric frame. (b) Object-centric frame.

Fig. 2: Visualizing the end-effector positions for all demonstrations under different coordinate frames. Each black dot is an xyz-position of the end effector in one step. We highlight one trajectory, which starts with red dots and ends in blue.

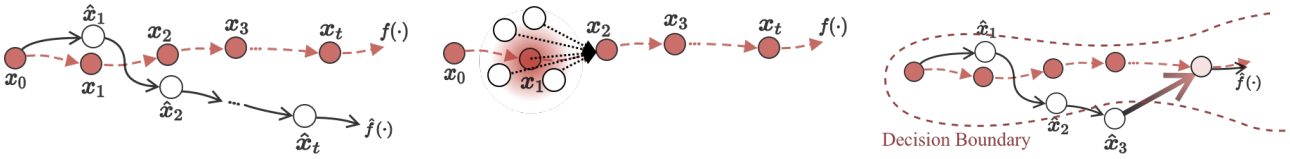
In manipulation, changing the frame of reference can significantly change the distribution of trajectories (Fig. 2). We could choose a *robot-centric* frame, where the robot base is the origin, or an *object-centric* frame [23], where the object location is the origin. The change of frame preserves the relative transformation between the end-effector and the object and is therefore an invariant operator. We propose that using an object-centric frame can reduce the covariate shift and improve the policy generalization, especially for fine manipulation. The transformation to an object-centric frame would result in a denser distribution of trajectories near the origin where the object is located, increasing data support for this critical region that determines grasping success. Using an object-centric frame also allows the policy learned to be invariant to the translation of object location. This makes the learned policy more sample efficient when generalizing to novel object locations.

B. Noise: Generating Synthetic Corrective Labels

Although the transformation technique we use improves the agent's success rate, we still observe significant deviations during test time that result in task failure (Fig. 3a). This is understandable because machine learning algorithms generally need exponentially more data for progressive improvement [22]. Instead of naively collecting more data, we introduce corrective action labels that can help the agent recover from deviations. For example, Venkatraman *et al.* [24] rolled out trained agents, collected their deviation states and used model-predictive control to generate corrective labels to go back to the demonstrated trajectory. Unfortunately, models sufficiently accurate for fine manipulation can be challenging to build.

We propose to generate *synthetic* corrective labels by injecting noise into the collected demonstration *states* ("deviated state") and reusing the collected action ("corrective labels"), thus not requiring access to an expert or a model. Unlike DART and DAgger, which emphasize collecting corrective labels for the states that the agent will visit during rollout (test state distribution), we hypothesize that we do not need to match the deviated states' distribution accurately.

[1] We injected an independent Gaussian noise to each joint. Though 95% of the noise resulted in at most 0.35° deviation per joint, it lowered the expert success rate by 18% and forced the expert to spend more time completing each trajectory.



(a) Covariate shift: A learner roll out (black) deviates from the demonstration (red) and error accumulates.

(b) Inject noise into the collected states and reuse the collected action as synthetic corrective labels.

(c) Use non-parametric methods (like a k-NN) to return the agent to proper region when deviations occur.

Fig. 3: Prevent error escalation in imitation learning.

Instead, we need to collect enough corrective labels to *cover* the deviated states' distribution. Since we can generate labels for free without burdening an expert, we choose to generate labels for randomly sampled deviated states, thus simplifying the selection of states for which to generate synthetic corrective labels. Fig. 3b shows an example where we sample states around a demonstrated state and reuse the demonstrated action as synthetic corrective labels.

Researchers have injected noise [25] into problems that reduce a high-dimensional input to a low-dimensional output, e.g., for classification [26] and object recognition in visual and language domains [27]. In these works, such tasks are invariant under a wide variety of transformations [28]. However, our robotic manipulation task has *low-dimensional* states and actions, where the mapping learned may not be invariant to the noise. We provide two insights to justify why injecting noise can still be desirable.

First, we apply a *small* amount of additive Gaussian noise to the demonstration state instead of a large amount that could pollute the data by mapping a state to a detrimental action. Inspired by [29], which showed the effectiveness of noise injection for autoencoders by carefully tuning the magnitude of the noise, we generate Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma)$ to add to the collected states, where σ is the covariance of the noise. For simplicity, we correlate the noise size σ with the variance of the data.

Second, because of the structure of our problem, the collected action can serve as the corrective label for the noise-injected deviated state. Our state and action representations both include the end-effector pose. Therefore, when an agent starts drifting from a demonstrated trajectory and enters a deviated state, our algorithm can teach it to return to the original trajectory by reusing the same action label. Injecting noise can also ensure the learned policy is smooth, which is desired since we assume the actions are Lipschitz continuous w.r.t the states.

C. Ensemble: Following the Expert Advice

We can reduce error accumulation at unseen states by choosing methods that more effectively recover the action distribution independent of the states. A neural network's optimization objective is limited to its training data and will not necessarily generalize well to unseen inputs [30]. In contrast, non-parametric methods generate test outputs by combining the training data, their predictions must come from the training data and are therefore constrained [31]. e.g., a k-nearest neighbor (`k-NN`) agent will not cause the robot to move its joint positions beyond the interpolation of its training data.

As an example, we use `k-NN` in conjunction with behavior cloning (`BC`). Specifically, our agent follows the `k-NN` predicted action if the query state deviates from the training data (Fig. 3c).

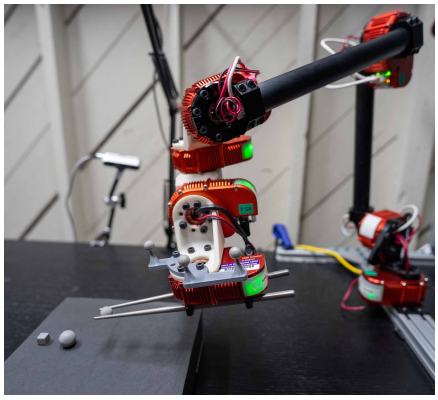
By using the `k-NN` method, we are *forcing a known action* to a new unseen state during test time to ensure the action distribution during training and testing will match. For our manipulation task, the state and action both include the robot's end-effector pose. Sending a *known action* is equivalent to sending the agent to *a known state*, implicitly reducing the agent's deviation from training data, thus reducing covariate shift. However, nonparametric method's performance is subject to its distance function, which can be difficult to design for high-dimensional data.

The distance function for non-parametric methods serves two purposes: (1) to evaluate the proximity of a query to the stored data points; and (2) to weight and combine the expert labels. Our key observation is that (1) requires only a rough estimate of the distance to decide whether a query state is far from the training data, and (2) needs a carefully tuned distance function to assign weights to expert labels. Therefore, we propose to use a simple decision tree to invoke a `k-NN` agent *only* when the distance of the query state is far from its nearest neighbors and invoke a behavior cloning neural network agent otherwise. By invoking `k-NN` only when the agent is far away, we bypass the need to carefully design a distance function for it, favor `BC`'s scalability with data when we are inside the training data distribution, and rely on `k-NN` to correct the agent's deviation when we are outside the training data distribution. We only explore `k-NN` to serve as an example and believe that other non-parametric methods that select actions from data-supported states could work similarly well.

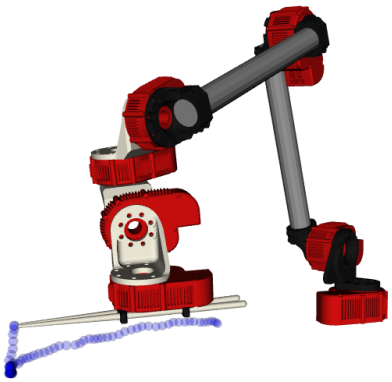
III. Experiments

A. Experimental Setup

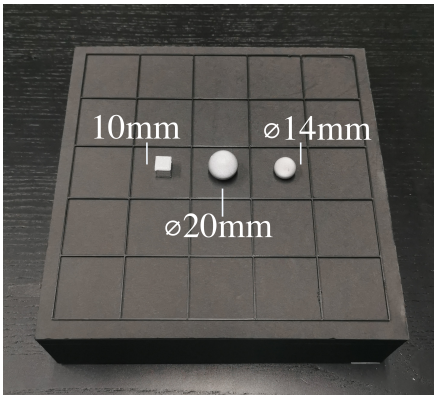
We built a 6-DOF robot (Fig. 4a) equipped with a pair of chopsticks as its end effector in order to develop algorithms that control the chopsticks to pick up challenging objects: a cube with a 1 cm edge length, a ball with a 2 cm diameter, and another ball with 1.4 cm diameter, as shown in Fig. 4c. We use Optitrack to track the locations of the object to grasp. The kinematic model for our inexpensive hardware is not highly accurate since the robot is assembled from parts with joints that are not strictly rigid. Even with the best



(a) Robot platform.



(b) Example demonstration.



(c) Evaluation.

Fig. 4: Experiment setup.

calibration, inaccuracies still accumulate along robot links and result in position errors ranging from 1 mm to 6 mm at the robot's end effector. This implies that the difference between the calculated chopstick tip position and its actual position is comparable to the radius of the small objects used in our experiments. For each object, we collected 500 trajectories from an expert teleoperating the robot to pick up the object (Fig. 4b). The data collection setup follows our previous work [9] and is summarized in Appendix V-A.

Our agent had access to the tracked location of the objects and the robot's end-effector pose. We defined success as *grasping* the objects using chopsticks, *lifting* them above the workstation, and *holding* them in the air for 1 s. We evaluated the performance of each method on each object by computing the success rate over 25 trials. During evaluation, we divided the square workstation plate into a 5×5 grid (Fig. 4c) and placed the object in the center of each grid cell to ensure effective coverage over the entire workspace. See Appendix V-A for more details.

B. Experimental Procedure

We compared our methods in Section II with human demonstrations during teleoperation (`Expert`) and a replay of the successful demonstrations (`Replay`). `Replay` tests the *repeatability* of our hardware. We chose successful demonstrations, placed objects at *exactly the same locations* used during data collection, and replayed the demonstrations to see if the robot could pick up the objects.

We used two baselines. The first is a parametric method, `BC+RobotC`, a neural-network based behavior cloning agent that uses the default robot-centric frame. The second is a non-parametric method, `k-NN+RobotC`, which is a k-nearest-neighbor agent that also uses the robot-centric frame.

We evaluated three methods as described in Section II: (1) using the object-centric frame to train behavior cloning and the k-nearest neighbors agents, `BC+ObjC` and `k-NN+ObjC`, respectively, (2) injecting a small amount of Gaussian noise into the behavior cloning agent, `BC+ObjC+Noise`, and (3) combining the parametric method `BC+ObjC+Noise` and non-parametric method `k-NN+ObjC` via a decision tree model, denoted as `Ensemble`. Implementation details are shown in the Appendix V-B.

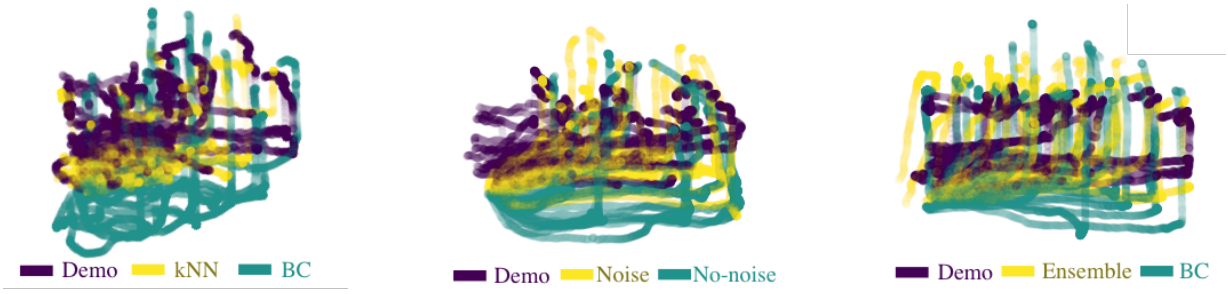
Method	Cube	Ball⌀20 mm	Ball⌀14 mm	All
`Expert`	100	80	68	82.7
`Replay`	100	80	80	86.7
`BC +RobotC`	84	16	12	37.3
`BC +ObjC`	92	16	24	44.0
`BC +ObjC+Noise`	92	76	48	72.0
`k-NN +RobotC`	64	28	8	33.3
`k-NN +ObjC`	84	64	12	53.3
`Ensemble`	**96**	**84**	**60**	**80.0**

TABLE I: Percentage success rates evaluated over 25 trials.

IV. Results

A. Success Rates for Fine Manipulation

The experimental results are shown in Table. I, and the best performers in each column are highlighted. Our parametric method baseline, `BC+RobotC`, and nonparametric method baseline, `k-NN+RobotC`, had relatively low success rates. However, the causes of their failures differ. `BC+RobotC` has difficulty picking up objects that are placed farther away from the robot. The agent tends to reach towards the wrong location after moving over a long distance to approach the object, highlighting the covariate shift's impact. In contrast, the `k-NN+RobotC` agent's poses look more similar to expert demonstrations. However, its trajectories are not smooth and sometimes end abruptly on top of the object without picking it up. This occurs because `k-NN` does not guarantee a smooth policy function; even after careful tuning of the distance function, it was challenging to eliminate the jerky motions. `k-NN`'s sudden stops are due to direct imitation of the training data. During demonstration, the human expert often slowed or even paused their movements around the object, adjusting the approaching pose before closing the chopsticks and lifting the object. The distance function we chose fails to select and mix the more relevant action labels. This confirms the sensitivity of `k-NN` to its distance function.



(a) `k-NN+ObjC` versus `BC+ObjC`. (b) `BC+ObjC` versus `BC+ObjC+Noise`. (c) `Ensemble` versus `BC+ObjC+Noise`.

Fig. 5: Comparing the state distributions as a proxy of covariate shift for trained agents. Each dot is a state in the agent's roll out. Green states are farther away from the demonstrations (purple), indicating that their corresponding agent suffers from more covariate shift than the yellow agent.

Transforming to the `ObjC` frame improved the success rates for `k-NN` and `BC` by 20% and 6.7%, respectively. `k-NN` becomes less likely to generate jerky motions or stop since it benefits from the increased data support. `BC` still suffers from covariate shift, but the agent has a higher likelihood of reaching towards the object due to denser data distribution near it.

Injecting noise to `BC+ObjC` during training increases its success rate by 28%. When the items are *close* to the robot, the agent has an almost 100% success rate picking up even the most challenging item. For objects that are far away, the robot sometimes picks up the object by successfully reaching the location; at other time, it ends up merely rotating the chopsticks.

Using a decision tree to combine our best parametric method (`BC+ObjC+Noise`) and non-parametric method (`k-NN+ObjC`) yields the highest performing agent that achieves near-expert performance. During test time, if a state's distance to its nearest neighbors exceeds a threshold, the agent triggers the non-parametric method to bring the state back. We observe that almost all rollouts trigger the non-parametric method at least once. We observed that the `Ensemble` agent can reach an object in a way similar to poses demonstrated by the expert, no matter how far it is placed. Failures occasionally occur as the agent misses the grasping point by some sub-mm error.

B. Covariate Shift Across Methods

To gauge the covariate shift for different agents, we visualize the distributions of their test states. We collect 25 rollouts from each agent, record the robot-visited states, and plot the state distribution after dimensionality reduction using Principal Component Analysis (PCA), as shown in Fig. 5. First, we observe that `BC` encounters more covariate shift than `k-NN`, i.e., that states visited by `k-NN` are closer to the demonstrated states, confirming that a better matching of action distribution will lead to a better match of state distribution. Second, injecting noise into `BC` results in less covariate shift than no noise, verifying that noise injection can provide effective correcting labels. Third, the `Ensemble` model that combines `BC` with `k-NN` has less covariate shift than using `BC` alone.

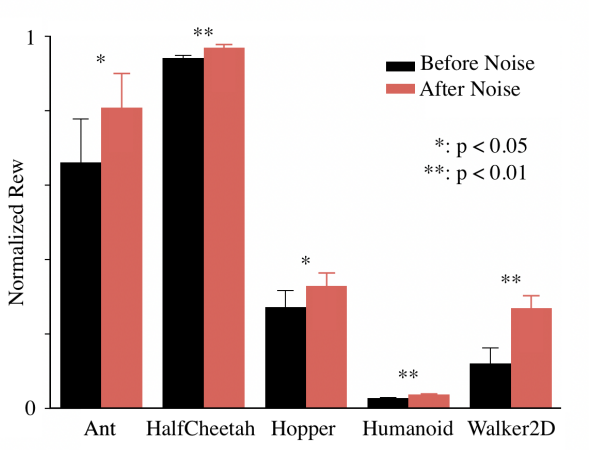


Fig. 6: Performance comparison before and after noise injection. For each MuJoCo environment and each condition, (before or after noise), we trained 5 agents using consecutive random seeds. The performance difference is statistically significant under paired T-tests.

C. Noise Injection: Validation through MuJoCo Environments

We apply the noise injection method to MuJoCo simulated environments [32] to test the method's generality. We use demonstration data from [33], train 5 behavior cloning agents under consecutive random seeds as baselines, and train another 5 agents with noise injection for comparison. Figure 6 compares performances before and after noise injection. A paired T-test shows that $p < 0.05$ for all environments. There is strong evidence that, on average, noise injection improves the imitation learner.

Though the performance gains for some simulated environments are not as significant as those we see for our real robot, we think the difference may be due to the demonstration source. We use "real" human data for our robot experiment versus the "synthetic expert demonstration" generated by a reinforcement learning (RL) agent for the MuJoCo tasks. Human experts are known to exhibit multi-modal behaviors during demonstrations, whereas trained RL agents tend to have single modes in their reaction [34]. Given that noise injection improves the success rate for our physical

robot task by a considerable 28%, further inquiry is needed to determine if noise injection is better at enhancing learning from multi-modal demonstration data.

V. Discussion

We leave some topics for future work. During noise injection, for simplicity, we experiment only with independent multivariate Gaussian noise with a fixed size of covariance. It is worth exploring how to formalize the bounded noise and analyzing how different task domains may benefit from different noise shapes. For the ensemble model, future work could explore an alternative way to switch between `k-NN` and `BC` agents in the `Ensemble` model, perhaps by learning a threshold condition from the data.

Our work critically depend on two key assumptions. First, to increase data support by applying an invariant operator, we assume the existence of a *critical* region that demands more data support. Second, to reuse collected action labels and leverage a nonparametric method to generate corrective labels, a more accurate match of action distribution should lead to a more accurate match of the states. The assumption holds if a part of the state and action representation is directly connected, e.g., the robot state contains its joint position, and the robot command accepts the target joint position. The assumption does not hold, for example, if the robot is torque-controlled; in these cases, further exploration on how a learner can generate synthetic corrective labels is needed.

Nevertheless, our proposals do not assume access to a model or an interactive expert and are therefore more easily applicable to fine manipulation tasks. Compared to DAgger and DART, which collect corrective labels from experts, we can generate synthetic corrective labels for free. Because of the relatively lower cost of doing so, we generate labels for randomly sampled state distributions that *cover* the deviated state distribution without accurately *matching* it. Though our proposals focus on a non-interactive setting, they can directly transfer to an interactive one.

We choose model-free imitation learning because an *accurate* model for fine manipulation is rare. However, it remains to be seen how to leverage an *inaccurate* model in imitation learning. This work is but our first step towards exploring general-purpose autonomous fine manipulation using simple tools. We look forward to extending it by combining model-free and model-based methods to manipulate a more diverse set of hard-to-grasp small objects.

Appendix

A. Experimental Setup

a) Robotic Testbed: We use the end-effector (EE) pose to describe the robot's state, which is an 8D vector containing (1) the x-y-z position of the bottom chopstick tip, (2) a quarternion representation of the rotation of the chopsticks, and (3) the opening angle of the last joint. We command the robot by sending a target end-effector pose at 100Hz, using an Inverse Kinematics solver to translate to joint positions and running a PID controller at 500Hz to move each joint.

b) Calibration Improved Performance: The default model and controller for our hardware were not highly accurate. The average EE position error was $10\,$mm. After careful calibration, we reduced this error to $4\,$mm. Initially, even a well-tuned controller had low success rates for picking up a cube and small ball during replay (90% and 15%, respectively). We implemented a custom PID controller and gain-tuning to achieve 100% and 80%, respectively.

c) Demonstrations: We collect the demonstration at 100 Hz to match the test scenario. Each trajectory contains an average of 600 (state, action) pairs. The state is a 11-D vector containing the robot's state and the object's tracked x-y-z position. The action is the target end-effector pose. During each trajectory, we initiate the robot around a fixed home configuration and place the object at a random location across the workstation. One expert user collect all trajectories to reduce multi-modal behavior that might interfere with learning (e.g., picking up object using different strategies). We remove failed trajectories and keep only the 500 successful ones.

B. Implementation Details

a) `BC`*:* We trained a two-layer fully connected neural network of size 64×32 with $ReLU$ activation. It outputs the 8D target end-effector pose. To compute its loss, we divided the 8D pose to position, rotation, and opening angle and computed the loss for each component using the mean squared error or the rotation difference. We then used a weighted linear combination to sum the components' losses to a 1D loss. The weights are tunable parameters.

b) `k-NN`*:* We used the last 3 end-effector poses and the current object location as input to the `k-NN` agent. We specify its distance function to be similar to the `BC` loss function but use a different set of weights.

c) `Noise`*:* During training of `BC` agents, instead of optimizing $\sum_{i \in \text{ batch}}[f(x_i) - a_i]$, where x_i is the state and a_i is the action, we sample 20% of the data in each batch and replace the state x_i with $x_i' = x_i + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma)$. σ is a diagonal matrix whose diagnoal entries are $\eta\hat{\sigma}$. We choose a fixed noise magnitude, η, for all dimensions of the state and $\hat{\sigma}$ is the variance of each dimension of the state. Empirically, $\sigma = \eta$ also achieves comparable performance.

d) `Ensemble`*:* Given that `k-NN` yields the nearest neighbors for a state and the corresponding distances, we set a threshold parameter α such that the agent follows `BC` iff $\sum(d_i)/k < \alpha$, where d_i the distance to the i-th closest neighbor. Further details are in [35].

Acknowledgement

Research reported in this publication was supported by the Eunice Kennedy Shriver National Institute Of Child Health & Human Development of the National Institutes of Health under Award Number F32HD101192. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health. This work was also (partially) funded by the National Science Foundation IIS (#2007011), National Science Foundation DMS (#1839371), the Office of Naval Research, US Army Research Laboratory CCDC, Amazon, and Honda Research Institute USA.

REFERENCES

[1] "Gross and fine manipulation." [Online]. Available: https://www.bls.gov/ors/factsheet/gross-and-fine-manipulation.htm#

[2] M. T. Mason, A. Rodriguez, S. S. Srinivasa, and A. S. Vazquez, "Autonomous manipulation with a general-purpose simple hand," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 688–703, 2012.

[3] H. Sakurai, T. Kanno, and K. Kawashima, "Thin-diameter chopsticks robot for laparoscopic surgery," in *International Conference on Robotics and Automation*, 2016, pp. 4122–4127.

[4] R. A. Joseph, A. C. Goh, S. P. Cuevas, M. A. Donovan, M. G. Kauffman, N. A. Salas, B. Miles, B. L. Bass, and B. J. Dunkin, "Chopstick surgery: a novel technique improves surgeon performance and eliminates arm collision in robotic single-incision laparoscopic surgery," *Surgical Endoscopy*, vol. 24, no. 6, pp. 1331–1335, 2010.

[5] M. Ragupathi, D. I. Ramos-Valadez, R. Pedraza, and E. M. Haas, "Robotic-assisted single-incision laparoscopic partial cecectomy," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 6, no. 3, pp. 362–367, 2010.

[6] A. A. Ramadan, T. Takubo, Y. Mae, K. Oohara, and T. Arai, "Developmental process of a chopstick-like hybrid-structure two-fingered micromanipulator hand for 3-d manipulation of microscopic objects," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 4, pp. 1121–1135, 2009.

[7] B.-C. Chang, B.-S. Huang, C.-K. Chen, and S.-J. Wang, "The pincer chopsticks: The investigation of a new utensil in pinching function," *Applied Ergonomics*, vol. 38, no. 3, pp. 385–390, 2007.

[8] A. Yamazaki and R. Masuda, "Autonomous foods handling by chopsticks for meal assistant robot," in *ROBOTIK 2012; 7th German Conference on Robotics*. VDE, 2012, pp. 1–6.

[9] L. Ke, A. Kamat, J. Wang, T. Bhattacharjee, C. Mavrogiannis, and S. S. Srinivasa, "Telemanipulation with chopsticks: Analyzing human factors in user demonstrations," in *International Conference on Intelligent Robots and Systems*. IEEE, 2020.

[10] A. Billard and D. Grollman, "Imitation learning (of robots)," Springer, Tech. Rep., 2011.

[11] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[12] J. Ho, J. Gupta, and S. Ermon, "Model-free imitation learning with policy optimization," in *International Conference on Machine Learning*, 2016, pp. 2760–2769.

[13] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *International Conference on Robotics and Automation*. IEEE, 2009, pp. 763–768.

[14] J. S. Dyrstad, E. R. Øye, A. Stahl, and J. R. Mathiassen, "Teaching a robot to grasp real fish by imitation learning from a human supervisor in virtual reality," in *International Conference on Intelligent Robots and Systems*. IEEE, 2018, pp. 7185–7192.

[15] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *International Conference on Robotics and Automation*. IEEE, 2018, pp. 1–8.

[16] M. R. Cutkosky, *Robotic grasping and fine manipulation*. Springer Science & Business Media, 2012, vol. 6.

[17] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, 2019.

[18] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems*, 1989, pp. 305–313.

[19] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *arXiv preprint arXiv:1811.06711*, 2018.

[20] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[21] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[22] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, "Dart: Noise injection for robust imitation learning," *arXiv preprint arXiv:1703.09327*, 2017.

[23] M. T. Mason, S. S. Srinivasa, and A. S. Vazquez, "Generality and simple hands," in *Robotics Research*. Springer, 2011, pp. 345–361.

[24] A. Venkatraman, M. Hebert, and J. A. Bagnell, "Improving multi-step prediction of learned time series models," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[25] C. M. Bishop, "Training with noise is equivalent to tikhonov regularization," *Neural Computation*, vol. 7, no. 1, pp. 108–116, 1995.

[26] J. Sietsma and R. J. Dow, "Creating artificial neural networks that generalize," *Neural Networks*, vol. 4, no. 1, pp. 67–79, 1991.

[27] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[28] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*. MIT press Cambridge, 2016, vol. 1.

[29] B. Poole, J. Sohl-Dickstein, and S. Ganguli, "Analyzing noise in autoencoders and deep networks," *arXiv preprint arXiv:1406.1831*, 2014.

[30] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. USA: Prentice Hall Press, 2009.

[31] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[32] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[33] "University of california berkeley CS 285: Deep reinforcement learning," http://rail.eecs.berkeley.edu/deeprlcourse/, accessed: 2020-10-27.

[34] L. Ke, S. Choudhury, M. Barnes, W. Sun, G. Lee, and S. Srinivasa, "Imitation learning as f-divergence minimization," in *Proceedings of the Workshop on Algorithmic Foundations of Robotics*, 2020.

[35] "Additional supplementary details (google drive)." [Online]. Available: https://drive.google.com/file/d/1PsryvqkxB9bNuRzgoqYIvaIm0rLt0sd7/view?usp=sharing