Teach a Robot to FISH: Versatile Imitation from One Minute of Demonstrations

Siddhant Haldar* Jyothish Pari* Anant Rai Lerrel Pinto

* Equal contribution

New York University

fast-imitation.github.io

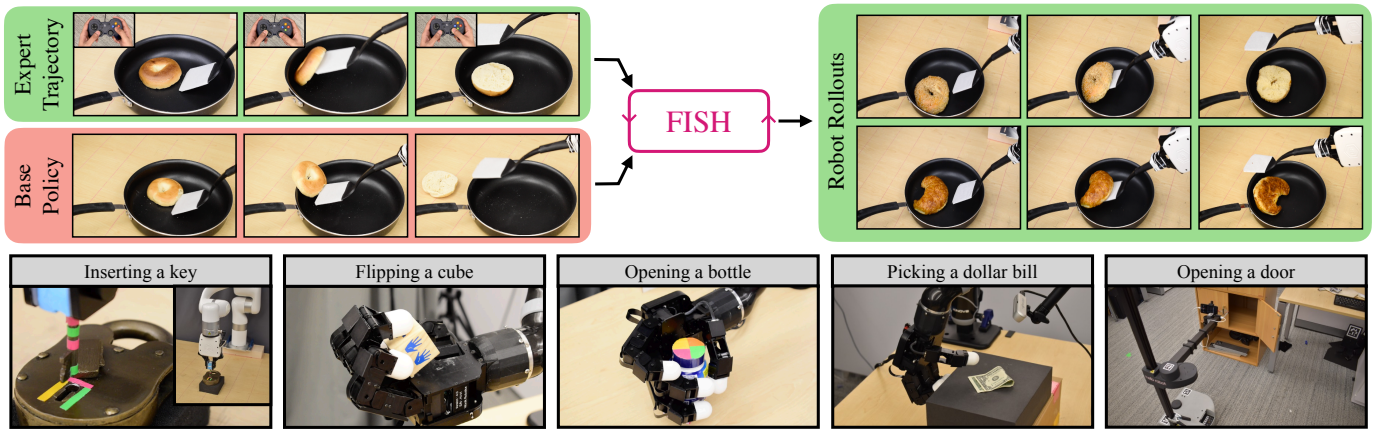

Fig. 1: Given less than one minute of expert trajectories by a human operator for a task, FISH can teach skills to a robot that adapt an offline imitated base-policy to new object configurations not seen in the demonstrations. This is done through an interactive RL procedure that improves the visual match between the robot's behavior and the demonstrated trajectory. Both the policy and matching function operate on visual observations. The versatility of FISH makes it compatible with a variety of robots (e.g. xArm, Allegro, Stretch) and tasks.

***Abstract*—While imitation learning provides us with an efficient toolkit to train robots, learning skills that are robust to environment variations remains a significant challenge. Current approaches address this challenge by relying either on large amounts of demonstrations that span environment variations or on handcrafted reward functions that require state estimates. Both directions are not scalable to fast imitation. In this work, we present Fast Imitation of Skills from Humans (FISH), a new imitation learning approach that can learn robust visual skills with less than a minute of human demonstrations. Given a weak base policy trained by offline imitation of demonstrations, FISH computes rewards that correspond to the "match" between the robot's behavior and the demonstrations. These rewards are then used to adaptively update a residual policy that adds on to the base policy. Across all tasks, FISH requires at most twenty minutes of interactive learning to imitate demonstrations on object configurations that were not seen in the demonstrations. Importantly, FISH is constructed to be versatile, which allows it to be used across robot morphologies (e.g. xArm, Allegro, Stretch) and camera configurations (e.g. third-person, eye-in-hand). Our experimental evaluations on 9 different tasks show that FISH achieves an average success rate of 93%, which is around 3.8× higher than prior state-of-the-art methods.**

I. Introduction

Imitation learning has proven to be among the most efficient tools to teach robots complex, dexterous and contact-rich skills. Its applications in robotics already span the fields of manipulation [29, 19], locomotion [44, 52], navigation [69, 30], and flying [18, 57]. Such imitation approaches are now gaining traction in directly learning from high-dimensional visual observations [25, 30, 35]. In broad strokes, visual imitation produces a policy that takes an image as input, and outputs actions that control the robot to perform desirable behaviors. Directly reasoning from images allows such methods to be generally applied as they circumvent the need for task-dependent estimation of state or design of features.

But, there is no free lunch. The generality of learning vision-based policies comes at the cost of needing a large number of demonstrations. MIME [59] uses 400 demonstrations per task, while robomimic [36] uses 200 demonstrations to train manipulation policies. This scale of data significantly hampers our ability to train multiple skills in reasonable amounts of time. Furthermore, collecting large amounts of demonstrations is physically and cognitively taxing on the human demonstrators due to the nature of available teleoperation frameworks [5]. Hence, getting imitation learning to work with few demonstrations is paramount for practical training of robotic skills.

To understand why imitation learning requires large amounts of data, let us take a look at one common paradigm – offline imitation. Methods in this class such as Behavior

Correspondence to: siddhanthaldar@nyu.edu, jyo.pari@nyu.edu

Cloning (BC) [47] or Nearest Neighbor retrieval (NN) [43] use a supervised learning objective to maximize the likelihood of demonstrated actions given observations in the demonstration. To ensure that the resulting policy is generalizable to varying factors in deployment (e.g. object configurations), the demonstration set used in training will need to span these factors of variation. Without sufficient coverage, which is only possible with large amounts of demonstration data, trained policies often suffer from distribution shift during deployment [54].

To address the large data requirements of offline imitation and instead imitate with few examples, a promising direction is to adapt policies that were trained offline with online RL [39, 25, 51]. The hope is that while the offline policy, trained with few demonstrations, would fail in deployment, online RL will allow the policy to improve and adapt to deployment scenarios. But how does the RL algorithm get the rewards needed for adaptation? Constructing a task-specific reward function is one possibility [51, 48]. However, this strategy may not be applicable in real-world scenarios where states of objects are hard to estimate or reward functions are hard to create.

In this work, we present Fast Imitation of Skills from Humans (FISH), a new technique for robotic imitation, where given only a minute of demonstrations (between 1 to 3 trajectories), a robot can learn visual policies that both solve the task and adapt to new object configurations through subsequent online training. FISH operates in two phases. First, a weak base policy is learned by offline imitation on the few demonstrations. Second, a residual policy [61, 28, 76, 2] is trained to produce corrective offsets to the weak policy. During online trial and error training, only the residual policy is updated, while the weak policy is queried as a black box model. This allows the use of non-parametric weak policies that are shown to be superior and more robust than parametric ones in low-data settings [43, 6, 5].

An important consideration in online policy learning is obtaining relevant rewards for robot behavior. Since we do not have access to task-specific reward functions, the rewards will need to be inferred from visual data. This is done by matching the visual observations from robot rollouts with the trajectory demonstrated by the human. The matching function uses fast approximations to Optimal Transport (OT) [14] to generate a matching score, which is proportional to the rewards. This procedure does not require explicit estimation of the state or any other object-centric representation.

We evaluate FISH on three different robot platforms that cover different morphologies, weak base policies, camera placements, and gripper types. Through an extensive study across 9 tasks, we present the following key insights:

1) FISH improves upon prior state-of-the-art work in online imitation [25, 12, 32] with an average of 93% improvement in success rate given 20 minutes of online interactions (Section III-D).
2) We find that FISH can generalize and adapt to a wide range of object configurations unseen in training (Section III-D).

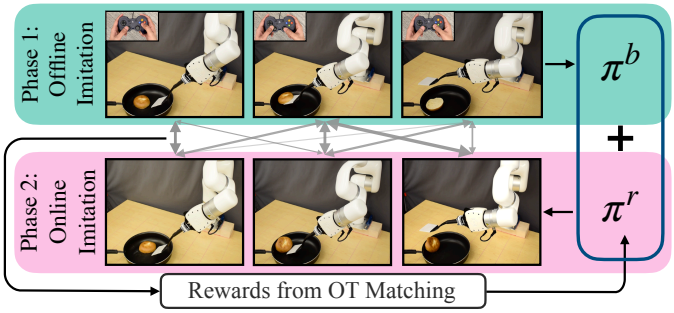

Fig. 3: A schematic of FISH. The first phase obtains a base policy through offline imitation from demonstrations. The second phase learns a residual model from online interactions

3) Ablations on different representation modules, adaptation strategies, and exploration strategies show that the design decisions in FISH are crucial for high performance (Section E).

Open-sourced code and videos of FISH can be found at: fast-imitation.github.io.

II. Approach

Given a few demonstrations for complex, contact-rich manipulation that covers a small subset of possible object configurations, we seek to learn a robot policy that can generalize to a larger set of configurations not seen during the demonstrations. To enable this, we propose Fast Imitation of Skills from Humans (FISH). FISH operates in two phases. In the first phase, a weak base policy is trained on the few demonstrations using supervised learning. This weak policy, while being poor in generalization, serves as a useful prior for subsequent adaptation. In the second phase, a residual policy is trained to adapt the base policy to new object configurations. This is done by RL on the robot with these configurations using visual trajectory matching scores as the reward signal.

A. Phase 1: Non-parametric base policy

The expert demonstrations are first used to derive an imperfect base policy π^b. In this work, we stick to non-parametric base policies owing to their proven robustness in the low-data regime [43, 6, 5] as compared to parametric alternatives such as Behavior Cloning (BC). We observe that different base policies perform differently across robots and thus, we employ two variants of non-parametric base policies in this work - an open-loop policy and closed-loop Visual Imitation through Nearest Neighbors (VINN) [43]. More details about these base policies have been provided in Section E3.

Visual representation learning: Since we operate in the visual domain, a BC policy is trained on the expert demonstrations and we use the encoder from the BC policy to encode the visual observations o into lower dimensional representations z. The encoded representation z is provided as an input to both the base policy π^b and the residual policy π^r. An ablation study comparing the use of such a BC encoder with other self-supervised learning techniques [24] as well as pretrained encoders [17, 71, 49, 40] is provided in Section E.

B. Phase 2: Online offset learning with IRL

Given the base policy π^b, we then train a residual policy π^r on top of the base policy through environment rollouts. Since we are operating without explicit task rewards, we obtain rewards using OT-based trajectory matching, as described in Section A. A standard RL optimizer utilizes these OT-based rewards r^{OT} to optimize the residual policy π^r by maximizing the cumulative reward from the final policy π^{FISH}. Similar to prior work [25, 12], we use n-step DDPG [34] as our RL optimizer, a deterministic actor-critic based method that provides high performance in continuous control [72].

Residual learning: In residual RL [61, 28, 76, 2], given a base policy $\pi^b : \mathcal{Z} \to \mathcal{A}$ with encoded representations $z \in \mathcal{Z}$ and action $a \in \mathcal{A}$, we learn a residual policy $\pi^r : \mathcal{Z} \times \mathcal{A} \to \mathcal{A}$ such that an action sampled from the final policy π is the sum of the base action $a^b \sim \pi^b(z)$ and the residual offset $a^r \sim \pi^r(z, a^b)$. In prior work, the base policy π^b is either a hand-crafted controller [61, 28] or a learned policy [2]. In this work, we use the non-parametric base policy π^b and learn a residual policy π^r using OT rewards to refine the action output by π^b.

OT-based reward maximization: For the RL algorithm, the rewards corresponding to an agent trajectory are computed using the OT-based approach described in Equation **??**. A visualization of OT rewards has been shown in Figure 5. These rewards are used to optimize the residual policy using the learning objective shown in Equation 1.

$$\pi^r = \underset{\pi^r}{\operatorname{argmax}}\, \mathbb{E}_{(z,a^b,a^r)\sim\mathcal{D}_\beta}[Q(z, a^b, a^r)] \tag{1}$$

Here, $Q(z, a^b, a^r)$ represents the Q-value from the critic used in actor-critic policy optimization. For an encoded representation z corresponding to observation o, $a^b \sim \pi^b(z)$ is the action from the base policy, and $a^r \sim \pi^r(z, a^b)$ is the action from the residual policy. The executed action a is a sum of a^b and a^r.

Stabilizing OT with representation learning: The OT rewards used for the IRL optimization are computed using the encoded representations. As a result, a changing encoder during training results in non-stationary rewards which makes the training prone to instabilities. In order to alleviate this issue, we fix the BC encoder obtained from the demonstrations and the OT rewards are computed using the representations from this fixed encoder. Section E5 shows that a fixed encoder improves stability resulting in superior performance.

Guided exploration for residual policy: In contrast to fine-tuning a base policy [25], applying offsets through a residual policy allows us to guide the exploration during online learning by injecting domain knowledge into the framework. For instance, if there is only a subspace of the full action space that we need to explore, our framework allows learning only the offsets for this subspace while keeping the base action along the remainder of the action space unaltered. We have provided ablation studies in Section E1 showing the advantage of such guided exploration. In addition to performance gains, constraining the offsets prevents the robot from going into undesirable positions and enables safer exploration during online learning (refer to Appendix F).

III. Experiments

Our experiments are designed to answer the following questions in detail: (1) How efficient is FISH for imitation learning? (2) How important is guided exploration for faster convergence? (3) How does the choice of base policy affect performance? (4) Are off-the-shelf pretrained encoders useful for online learning in a low-data regime? (5) How do additional implementation details affect FISH? (6) Does FISH generalize to new objects? (2-6 in Appendix E)

A. Experimental setup

We demonstrate the versatility of our algorithm by evaluating our approach on a suite of 9 tasks of varying difficulty across three different robot morphologies. We collect 1 minute of demonstrations (between 1 to 3 trajectories) for each task and allow a maximum of 20 minutes of online learning. For all tasks, we operate purely in the visual domain.

B. Robot setup and task descriptions

We evaluate our approach on 3 different robots - a Ufactory xArm 7 robot, an Allegro Hand, and a Hello Robot Stretch.

(a) **Ufactory xArm 7:** We use a xArm 7 robot with a two-fingered gripper for three tasks - key insertion, flipping a bagel, and peg in a cup. The observations are RGB images from a fixed external camera. For each task, the start position of the xArm is fixed and the object position is varied across trajectories. We use closed-loop VINN [43] as a base policy on the xArm. We provide one, two, and three expert demonstrations for the task of inserting a key, flipping a bagel, and inserting a peg in a cup respectively.

(b) **Allegro Hand:** We use a 4-fingered robotic hand with a 16-dimensional joint space. We study 3 dexterous manipulation tasks on the hand - cube flipping, bottle cap spinning, and dollar bill picking. The tasks have been designed to exhibit the need for dexterity to accurately manipulate the objects. The observations are RGB images from a fixed external camera. For each task, the start position of the hand is fixed and the object position is varied across trajectories during online training. We use an open-loop policy as a base policy on the hand and the demonstrations are collected using a virtual reality (VR) framework [5]. We use one expert demonstration for all tasks on the Allegro hand.

(c) **Hello Robot Stretch:** We use Hello Robot's Stretch to showcase our model's ability to interact with a realistic environment using a non-stationary robot. We perform three tasks using the Stretch Robot - door opening, drawer opening, and light switching. The observations are RGB images from an egocentric camera attached to the robot gripper. Hence, the camera viewpoint changes as the robot moves. For each task, the robot is initialized at a random position in front of the object. The demonstration we

TABLE I: We report the success rate on each task across different methods. The reported results are based on 5 evaluation trials on the Stretch and 10 evaluation trials each on the Allegro Hand and xArm. Candidate algorithms for base policies have been marked with [B].

Method	Stretch Robot			Allegro Hand			xArm		
	Door Opening	Drawer Opening	Light Switching	Cube Flipping	Bottle Cap Spinning	Dollar Bill Picking	Peg in a Cup	Bagel Flipping	Key Insertion
Open-loop [B]	0.2	0.2	0.2	0.1	0.0	0.2	0.1	0.1	0.3
VINN BC [B]	0.2	0.2	0.2	0.1	0.0	0.1	0.3	0.3	0.3
VINN ImageNet [B]	0.2	0.0	0.2	0.1	0.0	0.1	0.3	0.0	0.3
BC [B]	0.2	0.0	0.0	0.0	0.0	0.0	0.5	0.3	0.3
ROT	0.0	0.0	0.6	0.0	0.0	0.0	0.5	0.5	0.6
RDAC	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0
FISH (Ours)	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.8**	**0.9**	**0.9**	**0.8**

collect has the robot centered with respect to the handle of the door and drawer and the light switch. We used an open-loop policy as a base policy. We use one expert demonstration for all tasks on the Stretch robot.

For each task, we vary the position of the object or the robot at the start of each episode. All methods are evaluated on the same initial object or robot configurations, shown in Figure 7.

C. Baseline algorithms

We now describe the various imitation learning algorithms, both offline and online, used in this work.

(a) **Open-loop:** In settings where we have one demonstration, an open-loop policy copies the actions performed by the expert at each step of the trajectory. Though this yields robust performance when the object and robot's positions match the demonstration, it yields poor performance on any variations of the task.

(b) **Behavior Cloning (BC):** This refers to the behavior-cloned policy [47] trained on expert demonstrations.

(c) **Closed-loop VINN:** In closed-loop VINN [43], each visual observation in the demonstration is encoded into a representation. During rollouts, the k-Nearest Neighbors (kNN) algorithm is used to match to the k closest observations, and the action is computed using Locally Weighted Regression (LWR) [7] on the actions of the matched observations. In this work, we use a BC encoder for obtaining visual representations.

(d) **ROT:** ROT [25] is an IRL algorithm that finetunes a BC pretrained policy through online learning in an environment by leveraging optimal transport for reward computation. ROT gets around the "forgetting problem" in such a finetuning setting [39, 67] by using a soft Q-filtering based approach to prevent the actor from incorrectly deviating from the expert demonstration.

(e) **RDAC:** Discriminator Actor Critic (DAC) [32] is an adversarial imitation learning method [26, 64, 32]. DAC outperforms prior work such as GAIL [26] and AIRL [22]. RDAC is a DAC with a ROT-like regularization applied to it and has been observed to be a strong adversarial IRL baseline [25].

D. How efficient is FISH for imitation learning?

Performance of FISH on a suite of 9 real-world tasks across 3 different robots has been depicted in Table I. We observe that FISH outperforms prior work on all tasks. FISH significantly outperforms ROT [25], which is a method for finetuning a pretrained BC policy using online learning. This highlights the benefits of fixing a base policy as compared to modifying it during online finetuning. Further, aligned with results in Arunachalam et al. [6], we observe that BC performs poorly on the Allegro Hand owing to its high dimensional action space and limited demonstrations. This provides a case for using non-parametric base policies as opposed to parametric alternatives in such low-data regimes. Poor BC performance also affects online learning shown by the poor performance of ROT and is further shown in Section E3. We observe that while the learned BC policy is not robust enough to perform with high precision, the resulting representations are still sufficient for downstream fine-tuning (indicated by OT rewards shown in Figure 5). Empirically, we notice that BC is able to complete the coarse portions of the task such as reaching the object. However, the actions are often inaccurate indicating that the BC policy learned on top of the encoded representations is not precise enough.

IV. Conclusion

In this work, we present a new algorithm for fast imitation learning, FISH, that demonstrates improved performance compared to prior state-of-the-art work on a variety of real robot tasks across three different robot morphologies. We demonstrate that combining an imperfect base policy with a learned residual policy can enable performing precise tasks with one minute of demonstration collection and limited environment interactions. Further, we ablate over various design decisions of FISH, which shows the importance of learning stable representations, choosing the right base policy, and performing guided exploration. While powerful, we recognize that FISH has limitations (see Section E7).

Acknowledgments

We thank Sridhar Pandian Arunachalam, David Brandfonbrener, Zichen Jeff Cui, Venkatesh Pattabiraman, Ilija Radosavovic, and Chris Paxton for valuable feedback and discussions. This work was supported by grants from Honda, Meta, Amazon, and ONR awards N00014-21-1-2758 and N00014-22-1-2773.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

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

[3] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57 (5):469–483, 2009.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[5] Sridhar Pandian Arunachalam, Irmak Güzey, Soumith Chintala, and Lerrel Pinto. Holo-dex: Teaching dexterity with immersive mixed reality. *arXiv preprint arXiv:2210.06463*, 2022.

[6] Sridhar Pandian Arunachalam, Sneha Silwal, Ben Evans, and Lerrel Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation. *arXiv preprint arXiv:2203.13251*, 2022.

[7] Christopher G Atkeson, Andrew W Moore, and Stefan Schaal. Locally weighted learning. *Lazy learning*, pages 11–73, 1997.

[8] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[9] Edoardo Cetin and Oya Celiktutan. Domain-robust visual imitation learning with mutual information constraints. *arXiv preprint arXiv:2103.05079*, 2021.

[10] Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9329–9338, 2019.

[11] Samuel Cohen, Giulia Luise, Alexander Terenin, Brandon Amos, and Marc Deisenroth. Aligning time series on incomparable spaces. In *International Conference on Artificial Intelligence and Statistics*, pages 1036–1044. PMLR, 2021.

[12] Samuel Cohen, Brandon Amos, Marc Peter Deisenroth, Mikael Henaff, Eugene Vinitsky, and Denis Yarats. Imitation learning from pixel observations for continuous control, 2022. URL https://openreview.net/forum?id=JLbXkHkLCG6.

[13] Zichen Jeff Cui, Yibin Wang, Nur Muhammad, Lerrel Pinto, et al. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.

[14] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

[15] Marco Cuturi and Mathieu Blondel. Soft-dtw: a differentiable loss function for time-series. In *International conference on machine learning*, pages 894–903. PMLR, 2017.

[16] Robert Dadashi, Léonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal wasserstein imitation learning. *arXiv preprint arXiv:2006.04678*, 2020.

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[18] Yue Fan, Shilei Chu, Wei Zhang, Ran Song, and Yibin Li. Learn by observation: Imitation learning for drone patrolling from videos of a human navigator. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5209–5216. IEEE, 2020.

[19] Bin Fang, Shidong Jia, Di Guo, Muhua Xu, Shuhuan Wen, and Fuchun Sun. Survey of imitation learning for robotic manipulation. *International Journal of Intelligent Robotics and Applications*, 3:362–369, 2019.

[20] Arnaud Fickinger, Samuel Cohen, Stuart Russell, and Brandon Amos. Cross-domain imitation learning via optimal transport. *arXiv preprint arXiv:2110.03684*, 2021.

[21] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[22] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[24] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33: 21271–21284, 2020.

[25] Siddhant Haldar, Vaibhav Mathur, Denis Yarats, and Lerrel Pinto. Watch and match: Supercharging imitation with regularized optimal transport. *arXiv preprint arXiv:2206.15469*, 2022.

[26] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[27] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50 (2):1–35, 2017.

[28] 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.

[29] Edward Johns. Coarse-to-fine imitation learning: Robot manipulation from a single demonstration. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 4613–4619. IEEE, 2021.

[30] Haresh Karnan, Garrett Warnell, Xuesu Xiao, and Peter Stone. Voila: Visual-observation-only imitation learning for autonomous navigation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2497–2503. IEEE, 2022.

[31] Philip A Knight. The sinkhorn–knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30(1):261–275, 2008.

[32] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*, 2018.

[33] Sateesh Kumar, Jonathan Zamora, Nicklas Hansen, Rishabh Jangir, and Xiaolong Wang. Graph inverse reinforcement learning from diverse videos. *arXiv preprint arXiv:2207.14299*, 2022.

[34] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[35] Zhao Mandi, Fangchen Liu, Kimin Lee, and Pieter Abbeel. Towards more generalizable one-shot visual imitation learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2434–2444. IEEE, 2022.

[36] 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.

[37] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[38] 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.

[39] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[40] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[41] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[42] Georgios Papagiannis and Yunpeng Li. Imitation learning with sinkhorn distances. *arXiv preprint arXiv:2008.09167*, 2020.

[43] Jyothish Pari, Nur Muhammad Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021.

[44] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[45] Gabriel Peyré, Marco Cuturi, and Justin Solomon. Gromov-wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*, pages 2664–2672. PMLR, 2016.

[46] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[47] D Pomerleau. An autonomous land vehicle in a neural network. *Advances in Neural Information Processing Systems*, 1, 1998.

[48] Ilija Radosavovic, Xiaolong Wang, Lerrel Pinto, and Jitendra Malik. State-only imitation learning for dexterous manipulation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7865–7871. IEEE, 2021.

[49] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. *CoRL*, 2022.

[50] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Visual adversarial imitation learning using variational models. *Advances in Neural Information Processing Systems*, 34, 2021.

[51] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[52] Nathan Ratliff, J Andrew Bagnell, and Siddhartha S Srinivasa. Imitation learning for locomotion and manipulation. In *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pages 392–397. IEEE, 2007.

[53] Ievgen Redko, Titouan Vayer, Rémi Flamary, and Nicolas Courty. Co-optimal transport. *arXiv preprint arXiv:2002.03731*, 2020.

[54] Stéphane Ross, Geoffrey Gordon, and Drew 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, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[55] Maziar Sanjabi, Jimmy Ba, Meisam Razaviyayn, and Jason D Lee. On the convergence and robustness of training gans with regularized optimal transport. *Advances in Neural Information Processing Systems*, 31, 2018.

[56] Fumihiro Sasaki and Ryota Yamashina. Behavioral cloning from noisy demonstrations. In *International Conference on Learning Representations*, 2021.

[57] Fabian Schilling, Julien Lecoeur, Fabrizio Schiano, and Dario Floreano. Learning vision-based flight in drone swarms by imitation. *IEEE Robotics and Automation Letters*, 4(4):4523–4530, 2019.

[58] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *arXiv preprint arXiv:2206.11251*, 2022.

[59] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *Conference on robot learning*, pages 906–915. PMLR, 2018.

[60] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

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

[62] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[63] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

[64] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

[65] Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*, 2019.

[66] Sam Toyer, Rohin Shah, Andrew Critch, and Stuart Russell. The magical benchmark for robust imitation. *Advances in Neural Information Processing Systems*, 33: 18284–18295, 2020.

[67] Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, et al. Jump-start reinforcement learning. *arXiv preprint arXiv:2204.02372*, 2022.

[68] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

[69] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6629–6638, 2019.

[70] Huang Xiao, Michael Herman, Joerg Wagner, Sebastian Ziesche, Jalal Etesami, and Thai Hong Linh. Wasserstein adversarial imitation learning. *arXiv preprint arXiv:1906.08113*, 2019.

[71] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv:2203.06173*, 2022.

[72] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[73] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. *arXiv preprint arXiv:2008.04899*, 2020.

[74] Sarah Young, Jyothish Pari, Pieter Abbeel, and Lerrel Pinto. Playful interactions for representation learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 992–999. IEEE, 2022.

[75] Albert Zhan, Philip Zhao, Lerrel Pinto, Pieter Abbeel, and Michael Laskin. A framework for efficient robotic manipulation. *arXiv preprint arXiv:2012.07975*, 2020.

[76] Shangtong Zhang, Wendelin Boehmer, and Shimon Whiteson. Deep residual reinforcement learning. *arXiv preprint arXiv:1905.01072*, 2019.

APPENDIX

A. Background

Our work builds on several fundamental ideas in reinforcement learning, imitation learning and optimal transport. Here, we describe the most relevant background for FISH.

1) Reinforcement Learning (RL): We study RL as a discounted infinite-horizon Markov Decision Process (MDP) [8, 62]. For pixel observations, the agent's observation is approximated as a stack of consecutive RGB frames [37]. The MDP is of the form $(\mathcal{O}, \mathcal{A}, P, R, \gamma, d_0)$ where $\mathcal{O}$ is the observation space, $\mathcal{A}$ is the action space, $P : \mathcal{O} \times \mathcal{A} \to \Delta(\mathcal{O})$ is the transition function, $R : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ is the reward function, γ is the discount factor and d_0 is the initial state distribution. In this work, we use an actor critic based method to maximize the expected discounted sum of rewards. The rewards obtained through the OT computation can be used to optimize our policy through off-policy learning [32]. In this work, we use Deep Deterministic Policy Gradient (DDPG) [34] as our RL optimizer, which is an actor-critic algorithm that concurrently learns a deterministic policy π_ϕ and a Q-function Q_θ. Instead of minimizing the one-step Bellman residual as in vanilla DDPG, we use the n-step variant proposed by Yarats et al. [72] which has been successful on visual control problems.

2) Imitation Learning (IL): In imitation learning, the goal is to learn a behavior policy π^b from either an expert policy π^e or trajectories derived from an expert policy $\mathcal{T}^e$. In this work, we operate in a setting where the agent only has access to expert observational trajectories, i.e. $\mathcal{T}^e \equiv \{(o_t, a_t)_{t=0}^T\}_{n=0}^N$. Here, N refers to the number of trajectory rollouts and T denotes the episode length. We opt for this specific setting since obtaining expert or near-expert demonstrations is feasible in real-world settings [75, 73] and is in line with recent works in the area [25, 16, 26, 32].

3) Inverse Reinforcement Learning (IRL): IRL [41, 1] reformulates the IL problem in the RL setting by inferring the reward function r^e from expert trajectories $\mathcal{T}^e$. The inferred reward r^e is used to derive the behavior policy π^b using policy optimization. Prominent algorithms in IRL [32, 26] require alternating the inference of reward and optimization of policy in an iterative manner, which is practical for restricted model classes [1]. For compatibility with more expressive deep networks, techniques such as adversarial learning [26, 32] or optimal-transport [42, 16, 12] are needed. Adversarial IRL approaches infer a reward by learning a discriminator that minimizes the gap between expert trajectories $\mathcal{T}^e$ and behavior trajectories $\mathcal{T}^b$. Such a learning procedure results in non-stationary rewards r^e for the optimization of π^b which is prone to unstable training.

4) Optimal Transport for Imitation Learning (OT): To alleviate the non-stationary reward problem with adversarial IRL frameworks, a new line of OT-based approaches has been recently proposed [42, 16, 12]. Intuitively, the closeness between expert trajectories $\mathcal{T}^e$ and behavior trajectories $\mathcal{T}^b$ can be computed by measuring the optimal transport of probability mass from $\mathcal{T}^b \to \mathcal{T}^e$. During policy learning, an encoder f_ϕ transforms observations into informative state representations. Some examples of a preprocessor function f_ϕ are an identity function, a mean-variance scaling function and a parametric neural network. In this work, we use a parametric neural network as f_ϕ. Given a cost function $c : \mathcal{O} \times \mathcal{O} \to \mathbb{R}$ defined in the preprocessor's output space and an OT objective g, the optimal alignment between an expert trajectory $\mathbf{o}^e$ and a behavior trajectory $\mathbf{o}^b$ can be computed as

$$\mu^* \in \arg\min_{\mu \in \mathcal{M}} g(\mu, f_\phi(\mathbf{o}^b), f_\phi(\mathbf{o}^e), c) \tag{2}$$

where $\mathcal{M} = \{\mu \in \mathbb{R}^{T \times T} : \mu\mathbf{1} = \mu^T\mathbf{1} = \frac{1}{T}\mathbf{1}\}$ is the set of coupling matrices and the cost c can be the Euclidean or Cosine distance. The optimal alignment μ^* is subjected to the constraints $\sum_i \mu^*_{ij} = P(\mathbf{o}^e_i)\ \forall i$ and $\sum_j \mu^*_{ij} = P(\mathbf{o}^b_j)\ \forall j$. In this work, inspired by [12], we use the entropic Wasserstein distance [14] with cosine cost as our OT metric, which is given by the equation

$$\begin{aligned} g(\mu, f_\phi(\mathbf{o}^b), f_\phi(\mathbf{o}^e), c) &= \mathcal{W}^2(f_\phi(\mathbf{o}^b), f_\phi(\mathbf{o}^e)) \\ &= \sum_{t,t'=1}^{T} C_{t,t'}\mu_{t,t'} \end{aligned} \tag{3}$$

where the cost matrix $C_{t,t'} = c(f_\phi(\mathbf{o}^b), f_\phi(\mathbf{o}^e))$. Using Eq. 3 and the optimal alignment μ^* obtained by optimizing Eq. 2, a reward signal can be computed for each observation using the equation

$$r^{OT}(o^{\boldsymbol{b}}_t) = -\sum_{t'=1}^{T} C_{t,t'}\mu^*_{t,t'} \tag{4}$$

Intuitively, maximizing this reward encourages the imitating agent to produce trajectories that closely match demonstrated trajectories. Since solving Eq. 2 is computationally expensive, approximate solutions such as the Sinkhorn algorithm [31, 42] are used instead.

5) Soft Q-filtering: In ROT [25], a soft Q-filtering based approach is used for appropriately weighing the Q-value based actor-critic loss and the BC regularization loss in the objective function. The key idea behind this is to keep the online learning policy π^{ROT} close to the behavior policy on regions of the observation space where the behavior policy exhibits better performance than π^{ROT}. More precisely, given a behavior policy $\pi^{BC}(s)$, the current policy $\pi^{ROT}(s)$, the Q-function $Q(s, a)$ and the replay buffer $\mathcal{D}_\beta$, the BC regularization weight λ is computed as:

$$\lambda(\pi^{ROT}) = \mathbb{E}_{(s,\cdot)\sim\mathcal{D}_\beta}\left[\mathbb{1}_{Q(s,\pi^{BC}(s)) > Q(s,\pi^{ROT}(s))}\right] \tag{5}$$

This filtering strategy is inspired by Nair et al. [38], which employs a binary hard assignment instead of a soft continuous weight. In Table III, we provide an ablation study comparing the performance of FISH with and without such a regularization scheme. In FISH, since we learn a residual policy on top of a fixed base policy during online learning, such a regularization scheme must keep the final action, which is

the sum of the base action and the residual offset, close to the action in the expert demonstrations. To do this, we apply the regularization such that the offsets are regularized to stay close to zero when the base policy π^b is performing better than the policy π^{FISH}. Table III shows that applying such offset regularization harms the performance of our model. Empirically, we observe that such regularization drives the residuals to be a very small value close to zero which renders them ineffective in producing significant performance gains over the base policy.

B. Related Work

Imitation Learning (IL) IL [3, 27] has been shown to solve complex tasks in real-world environments. Approaches for IL include Behavior Cloning (BC) [47, 65] and Inverse Reinforcement Learning (IRL) [41, 1]. BC solely learns from offline demonstrations and has shown promising results in the presence of large diverse datasets [47, 63, 10, 56, 74]. Assistive tools and other teleoperation methods have allowed for more efficient data collection [73, 5]. BC has also been applied to tasks with a multimodal action distribution [21, 58, 13]. However, BC suffers on out-of-distribution samples [54] which renders it unsuitable for the low-data regime. Lately, there has been some work on utilizing non-parametric models to tackle such low data regimes with offline imitation [43, 6, 5]. A significant drawback of offline IL is that they do not provide any means for correcting the behavior on unseen observations. IRL provides a solution to this problem by learning a robust reward function through online interactions but suffers from sample inefficiency [32]. There has been some work on improving the sample efficiency of IRL [32, 22, 70, 64], with some visual extensions to these IRL approaches [25, 9, 66, 50, 12]. There has also been demonstration of such IRL approaches performing complex tasks on real robots [1, 25, 33].

Optimal Transport (OT) OT [68, 46] provides a tool for comparing probability measures while including the geometry of the space. In imitation learning, OT can be used to compute the alignment between a set of agent and expert observations using distance metrics such as Sinkhorn [14], Gromov-Wasserstein [45], GDTW [11], CO-OT [53] and Soft-DTW [15]. Many of these distance metrics have an associated IL algorithm - SIL [42] uses Sinkhorn, PWIL [16] uses greedy Wasserstein, GDTW-IL [11] uses GDTW, and GWIL [20] using Gromov-Wasserstein. Recent work by Cohen et al. [12] has demonstrated that the Sinkhorn distance [42] produces the most efficient learning among the discussed metrics and can be combined with offline pretraining to efficient perform complex tasks in the real world [25]. OT has also seen use in the field of computer vision [55, 4] to show improvements for Generative Adversarial Networks (GANs) [23]. In this work, we adopt the Sinkhorn metric for online learning and combine it with non-parametric IL approaches to perform precise tasks across three robot morphologies.

Residual RL for robotics Learning residuals through RL enables safe and robust online learning [61, 28, 76, 2]. Residual RL operates by applying offsets on top of a base policy. Prior works either use a hand-engineered controller [61, 28] or a policy learned from demonstrations [2] as the base policy. In this work, we resort to the latter and use non-parametric base policies obtained from one minute of expert demonstration. Prior works also assume the availability of task-specific rewards for learning the online policy. However, we differ from this and use OT matching to obtain rewards from the collected demonstration set.

C. Algorithm Details

1) Algorithm block: Algorithm 1 describes our proposed algorithm, Fast Imitation of Skills from Humans (FISH).

Algorithm 1 FISH: Fast Imitation of Skills from Humans

Require:
Expert Demonstrations $\mathcal{T}^e \equiv \{(o_t, a_t)_{t=0}^T\}_{n=0}^N$
BC pretrained Encoder f_{enc}
Base policy π^b
Replay buffer $\mathcal{D}$, Training steps T, Episode Length L
Task environment env
Parametric networks for RL backbone (e.g., the policy and critic function for DrQ-v2)

Algorithm:
Randomly initialize residual policy π^r and critic Q
for each timestep $t = 1...T$ **do**
  if done **then**
    $r_{1:L} = \text{rewarder}_{OT}(\text{episode})$ ▷ OT reward
    Update episode with $r_{1:L}$
    Add $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{a}_t^b, \mathbf{o}_{t+1}, r_t)$ to $\mathcal{D}$
    $\mathbf{o}_t = env.reset(),\ \text{done} = \text{False},\ \text{episode} = [\ ]$
  end if
  $\mathbf{z}_t = f_{enc}(\mathbf{o}_t)$
  $\mathbf{a}_t^b \sim \pi^b(\mathbf{z}_t)$ ▷ Compute base action
  $\mathbf{a}_t^r \sim \pi^r(\mathbf{z}_t, \mathbf{a}_t^b)$ ▷ Compute residual offset
  $\mathbf{a}_t = \mathbf{a}_t^b + \mathbf{a}_t^r$
  $\mathbf{o}_{t+1},\ \text{done} = env.step(\mathbf{a}_t)$
  episode.append$([\mathbf{o}_t, \mathbf{a}_t, \mathbf{a}_t^b, \mathbf{o}_{t+1}])$
  Update backbone-specific networks using $\mathcal{D}$
end for

2) Algorithm and training procedure: Our model consists of 3 primary neural networks - the encoder, the actor and the critic. During the BC pretraining phase, the encoder and the actor are trained using a mean squared error (MSE) on the expert demonstrations. Next, for finetuning, the pretrained BC encoder is loaded and is used to obtain the encoded observations. The actor and the critic are initialized randomly. The loaded BC encoder is fixed during online learning in order to stationarize the OT rewards which are computed on the encoded representations.

3) Actor-critic based reward maximization: We use a recent n-step DDPG proposed by Yarats et al. [72] as our residual RL backbone. The deterministic actor is trained using deterministic policy gradients (DPG) [60]. The critic is trained using clipped double Q-learning similar to Yarats et al. [72] in order

TABLE II: List of hyperparameters.

Method	Parameter	Value
Common	Replay buffer size	5000
	Learning rate	$1e^{-4}$
	Discount γ	0.99
	n-step returns	3
	Action repeat	1
	Seed frames	260 (xArm, Stretch), 200 (Allegro Hand)
	Mini-batch size	256
	Agent update frequency	2
	Critic soft-update rate	0.01
	Feature dim	50
	Hidden dim	1024
	Optimizer	Adam
FISH	Exploration steps	0
	DDPG exploration schedule	0.1
	Reward scale factor	10
ROT	Exploration steps	0
	DDPG exploration schedule	0.1
	Reward scale factor	10
	Fixed weight α	0.03
DAC	Exploration steps	2000
	DDPG exploration schedule	linear(1,0.1,2000)
	Gradient penalty coefficient	10

to reduce the overestimation bias in the target value. This is done using two Q-functions, $Q_{\theta 1}$ and $Q_{\theta 2}$. The critic loss for each critic is given by the equation

$$\mathcal{L}_{\theta_k} = \mathbb{E}_{(z,a^b,a^r)\sim D_\beta}\left[(Q_{\theta_k}(z,a^b,a^r) - y)^2\right] \forall\ k \in \{1,2\} \tag{6}$$

where $\mathcal{D}_\beta$ is the replay buffer for online rollouts and y is the target value for n-step DDPG given by

$$y = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \underset{k=1,2}{min} Q_{\bar{\theta}_k}(s_{t+n}, a^b_{t+n}, a^r_{t+n}) \tag{7}$$

Here, γ is the discount factor, r is the reward obtained using OT-based reward computation and $\bar{\theta}_1$, $\bar{\theta}_2$ are the slow-moving weights of target Q-networks.

4) Architecture Details: Our encoder takes in an 84×84 image as input and produces a 512-dimensional output. The encoder comprises 4 convolutional layers with a single linear layer. The actor takes in the encoded representation along with the action from the base policy and passes it through 3 linear layers to produce an action. The critic takes in the encoded representation, the action from the base policy, and the residual action and passes it through 3 linear layers to produce the Q-value.

5) Hyperparameters: The complete list of hyperparameters used in the paper has been provided in Table II.

D. Task description

In this section, we describe the suite of manipulation experiments carried out on a real robot in this paper.

(a) **Door opening:** The Stretch robot is supposed to open a closed door in a cupboard.
(b) **Drawer opening:** The Stretch robot is supposed to open a closed drawer in a kitchen.
(c) **Light switching:** The Stretch Robot is tasked with switching off a light switch that is in an ON position.
(d) **Cube flipping:** The Allegro Hand is tasked with flipping a cube placed on its palm by 90 degrees.
(e) **Bottle cap spinning:** The Allegro Hand is tasked with spinning a bottle cap by 270 degrees.
(f) **Dollar bill picking:** The Allegro Hand is tasked with sliding and picking up a bill placed on a platform.
(g) **Peg in a cup:** The xArm is tasked with placing a peg inside a cup which is moved to random positions on a table.
(h) **Bagel flipping:** The xArm is tasked with using a spatula to flip a bagel placed at random positions on a fry pan.
(i) **Key insertion:** The xArm is tasked with inserting a key inside the keyhole of a lock that is randomly placed on a table.

Evaluation procedure For each task, we obtained a set of random points where the object is placed (for xArm and the Allegro Hand) or the robot is initialized (for the Stretch

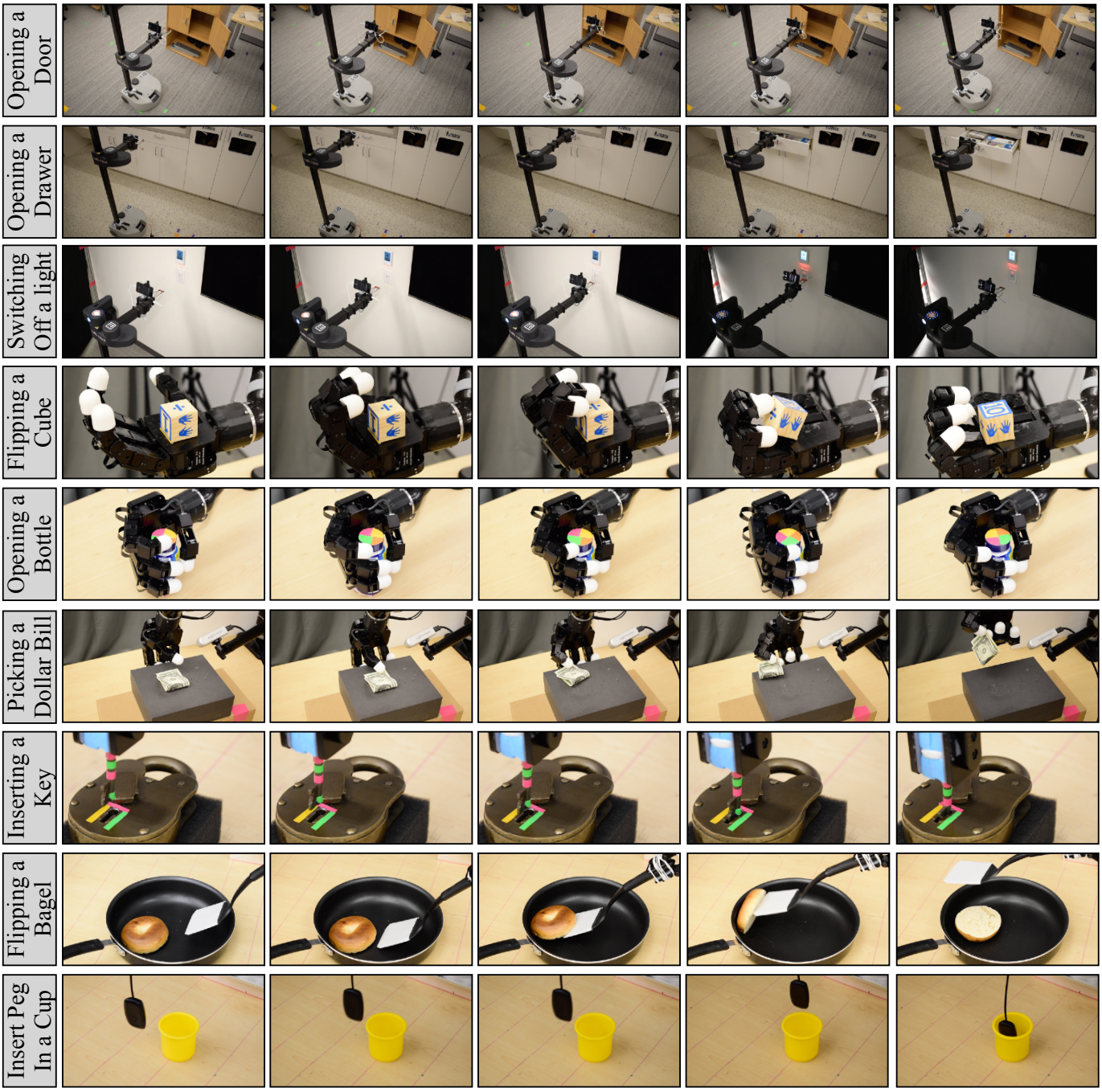

Fig. 4: Robot rollouts for the 9 tasks we evaluate FISH on.

robot) and evaluate FISH and all of the other methods on the same set of initializations. These initializations are different for each task based on the limits of the observation space of the task. For the tasks on the xArm and the Allegro Hand, we evaluate on a set of 10 different object positions while the tasks on the Stretch are evaluated on a set of 5 different robot initializations.

E. Additional Experiments

1) How important is guided exploration?: As opposed to finetuning a parametric model where any update to the model can affect all dimensions of the action space, learning residuals over a fixed-based policy allows us to guide our exploration. For instance, owing to its high dimensional action space, exploring along all dimensions of the action space in the Allegro Hand renders online learning ineffective. So depending on the base policy performance, we only apply

Fig. 5: An analysis of the values of OT rewards for different trajectories with respect to a given expert demonstration. The leftmost column depicts the visual demonstration, while the other columns each depict a trajectory rollout. Trajectories are sorted in increasing order of OT rewards from left to right. Raw OT scores can be visualized using the red-to-green color map.

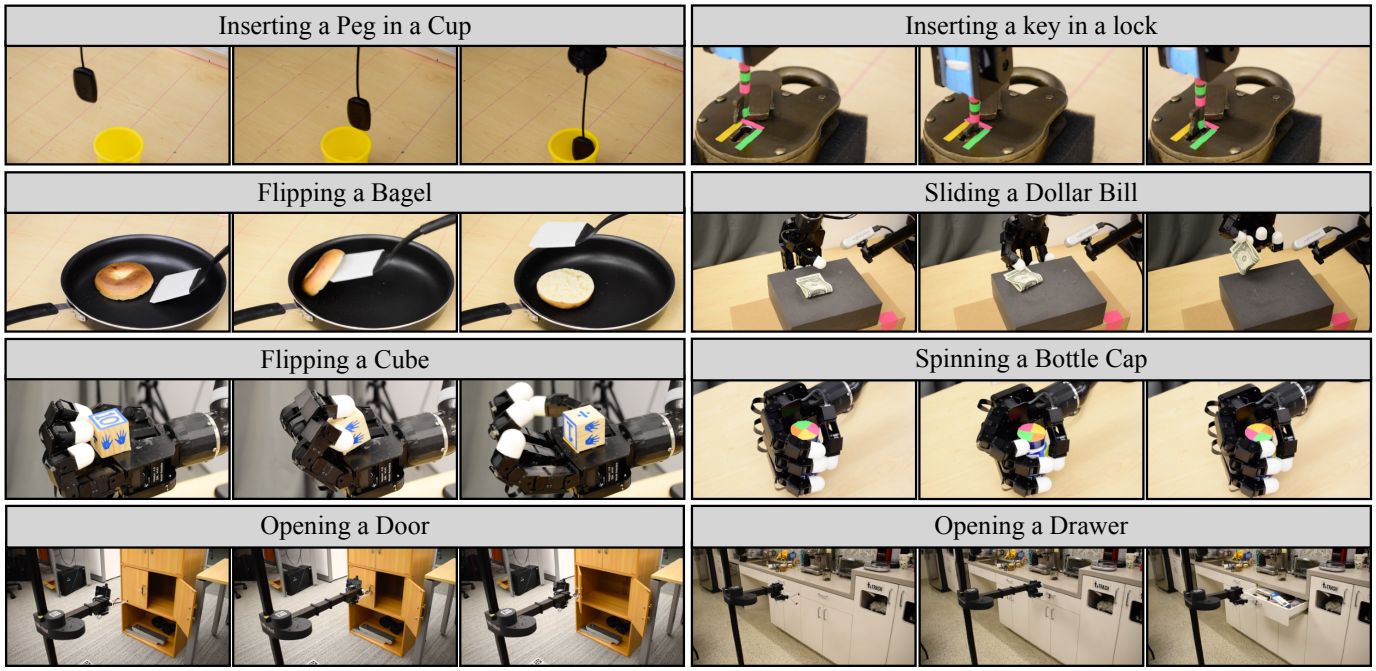

Fig. 6: A visualization of rollouts from FISH on a selected set of 8 tasks.

residuals along some dimensions while keeping the base policy unaltered along the remaining dimensions. Specifically, we divide our evaluations into three parts - guided, semi-guided, and unguided. For the bagel flipping task, we explore only along the Z-axis for the guided setting, along the XYZ axes for the semi-guided setting, and along both the XYZ axes and roll-pitch-yaw for the unguided setting. Figure 8 demonstrates the effectiveness of such guided exploration

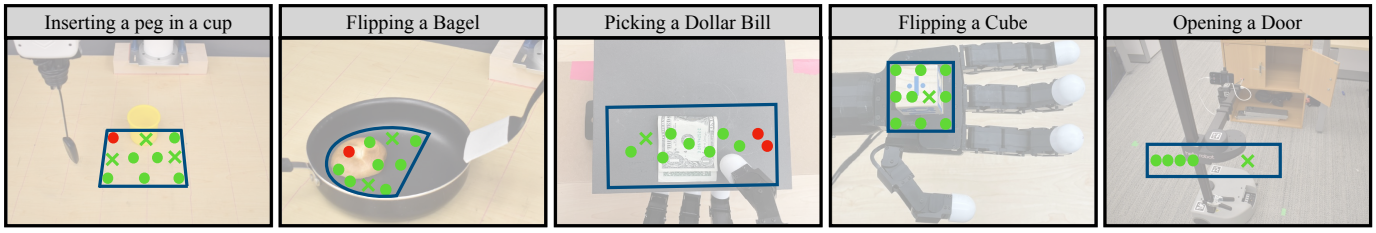

Fig. 7: Plot showing variation in object positions or robot initializations for selected tasks. The region of operation for each task is denoted by the blue box. × on the images indicate positions where the demonstrations are collected. The green marks indicate positions where FISH succeeds and the red ones indicate failure modes. As shown, FISH succeeds with varied object positions and initial robot configurations.

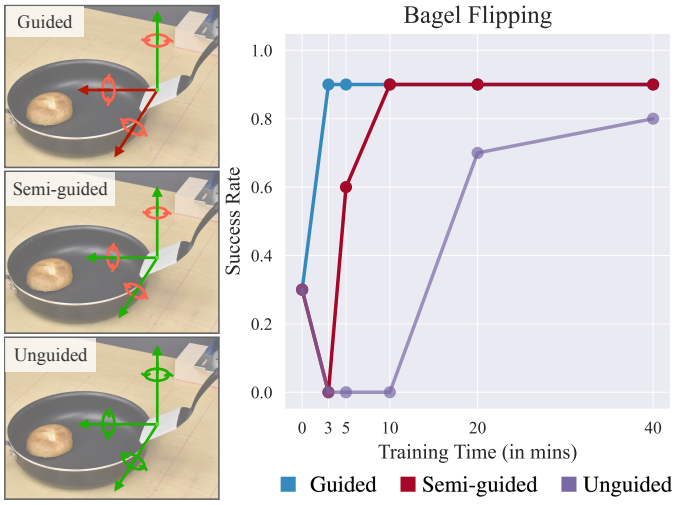

Fig. 8: Comparison between success rate for varied levels of guidance applied to the residual policy. On the left, we show the meaning of each level of guidance. In each scenario, the green axes denote the direction along which the offsets are learned.

TABLE III: Comparison between success rates for 10 trials on two tasks for FISH with and without adaptive regularization of the offsets.

Offset Regularization	Bagel Flipping	Dollar Bill Picking
✓	0.4	0.4
×	0.9	0.8

over the unconstrained alternative. Note that although guided exploration improves sample efficiency, unguided exploration with FISH still outperforms our strongest baselines in Table I.

2) Is adaptive regularization beneficial for learning offsets?: In Figure 8, we observe that though the base policy has non-zero performance in the bagel flipping task, applying offsets drives the performance to zero during the initial part of online learning. This is primarily due to the untrained offsets driving the agent to an observation unseen in the expert demonstration, thus, adversely affecting the base policy. Drawing inspiration from recent work that uses adaptive regularization to keep the online policy close to the base policy during the initial part of training [25], we adaptively regularize our residuals to stay close to zero using the same soft Q-filtering approach (see more details in Appendix A5). However, as observed in Table III, this harms the performance

TABLE IV: Comparison between success rates on 10 trials for our method with different base polices.

Method	Bagel Flipping	Dollar Bill Picking
IRL Scratch	0.0	0.0
Open-loop	0.1	**0.8**
BC	0.7	0.0
VINN (ImageNet)	0.0	0.0
VINN (BYOL)	0.0	0.0
VINN (BC Encoder)	**0.9**	0.0

of our model. Empirically, we observe that such regularization drives the residuals to be a very small value close to zero which renders them ineffective in producing significant performance gains over the base policy.

3) How does the choice of base policy affect performance?: To understand the effect of using different base policies, we compare the performance of FISH on four variants shown in Table IV. The finetuned ImageNet [17] encoder refers to a pretrained ImageNet encoder finetuned with BYOL [24] on the expert demonstrations. These experiments provide 3 key insights - (a) OT-based IRL without pre-training does not work well with few environment interactions, (b) self-supervised learning (SSL) methods such as BYOL do not work well in the low data regime, and (c) with a decent BC policy as in the case of bagel flipping, FISH can produce significant improvements on the base policy. However, using a non-parametric base policy such as VINN obtains a superior performance as compared to parametric alternatives.

4) Are pretrained encoders useful for online learning?: We compare the performance of FISH with the VINN base policy obtained from a variety of off-the-shelf encoders pretrained using self-supervised learning on large-scale datasets - ImageNet [17], MVP [71, 49] and R3M [40]. Table V shows that even though these encoders are trained on large-scale datasets, they do not perform well in this setting. In many cases, the performance is worse than our base policies. This is perhaps because the representations learned on Internet data may not transfer well to our suite of tasks. Further, this indicates that representations trained on in-domain data, even in the low-data regime, may perform better than training on large amounts of out-of-domain data.

5) How do additional implementation details affect FISH*?:* Table VI provides additional insights with regard to - (a)

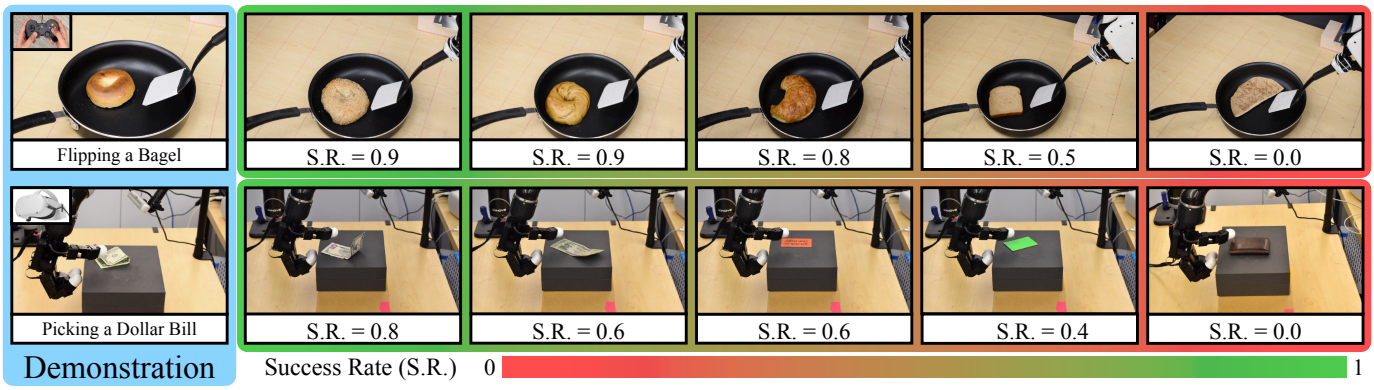

Fig. 9: Here we run FISH, showing one demonstration on the leftmost object and then training it on new objects. Success rates (S.R.) after 5 minutes of online learning are reported below the corresponding object.

TABLE V: Analysis of the performance of FISH using different pre-trained encoders.

Encoder	Bagel Flipping	Dollar Bill Picking
ImageNet	0.0	0.0
R3M	0.0	0.1
MVP	0.3	0.0
BC	**0.9**	**0.8**

TABLE VI: Ablation analysis on fixing encoders and conditioning on base actions during online learning.

Fix Encoder	Condition on base action	Bagel Flipping	Dollar Bill Picking
✓	×	0.6	0.1
×	✓	**0.9**	0.0
✓	✓	**0.9**	**0.8**

having a fixed encoder during online learning, (b) conditioning the residual policy on the base policy action. We observe that both of these techniques are necessary and dropping either of them adversely affects the performance of the algorithm.

6) Does FISH *generalize to new objects?:* We demonstrate the ability of FISH to generalize to different objects with varied appearances and dynamics. In Figure 9, we show this generalization for a representative task on the xArm and the Allegro Hand. We observe that the performance drops proportionally with the increase in variation. For instance, the xArm completely fails at flipping a flatbread which is considerably softer than a bagel and requires a different strategy to flip. Similarly, the hand fails to pick up a wallet that is thicker and more uneven than a dollar bill. However, even though the model fails in extreme cases, it succeeds at performing the task with a significant variation in visual and dynamic properties of the object. Figure 4 shows the generalization of FISH to different types of bread despite the demonstration being on a specific kind of bagel. Similarly, Figure 12 shows such generalization to different bills and cards despite the demonstration set comprising a specific dollar bill.

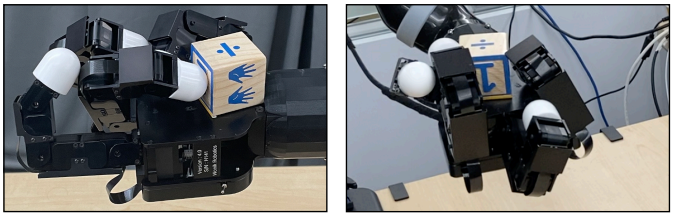

Fig. 10: Snapshots of the Allegro Hand while training an online policy using ROT. Here, we show that exploring along all action dimensions results in unsafe learning on the hand, driving it to undesirable positions and unnatural poses.

7) Limitations of FISH*:* To summarize our experiments, we showcase the effectiveness of our algorithm when operating in a low data regime with a limited budget for environment interactions. We demonstrate a significant improvement in performance as compared to prior state-of-the-art work and provide extensive ablations to justify our design choices. However, we recognize a few limitations in this work: (a) Since the OT-based rewards used to train the residual policy align the agent with the demonstrations, it relies on the demonstrator being an 'expert'. (b) We restrict ourselves to the visual domain which makes it difficult to perform precise tasks where the visual signals are not very prominent. For example, it is difficult to infer a keyhole spanning a minuscule portion of an image. A potential improvement along this line might result from embracing other modalities such as tactile sensing. (c) Our residual policy is randomly initialized. Pretraining the residual policy might help scale to more difficult tasks requiring more precise control.

F. Safe exploration using a guided residual policy

In FISH, we propose guiding the residual policy to learn offsets along a smaller subspace of the full action space. In other words, the residual policy is used to learn offsets only along certain action dimensions instead of the complete action space. In addition to providing performance benefits, as shown in Figure 8, we argue that such guided exploration also enables safer online learning. For instance, Figure 10 shows

Fig. 11: Generalization of FISH to different types of bread.

the Allegro hand during different stages of online learning while training ROT. ROT explores along all dimensions of the action space which drives to hand to undesirable positions, resulting in collisions between fingers and unnatural poses. This confirms that guided exploration of the action space, as proposed in FISH, enables safer online training of such systems.

Fig. 12: Generalization of FISH to different bills and cards.