Sample-Efficient Online Imitation Learning using Pretrained Behavioural Cloning Policies

Joe Watson[13], * **Jan Peters**[1234]
[1]Department of Computer Science, Technical University of Darmstadt
[2]Centre for Cognitive Science, Technical University of Darmstadt
[3]German Research Center for AI [4]Hessian.AI
{joe,jan}@robot-learning.de

Abstract

Recent advances in robot learning have been enabled by learning rich generative and recurrent policies from expert demonstrations, such as human teleoperation. These policies are capable of solving many complex tasks by accurately modelling human behaviour, which may be multimodal and non-Markovian. However, this imitation learning approach of behavioural cloning (BC) is limited to being offline, which increases the requirement for large expert demonstration datasets and does not enable the policy to learn from its own experience. In this work, we review the recent imitation learning algorithm coherent soft imitation learning (CSIL) and outline how it could be applied to more complex policy architectures. CSIL demonstrates that inverse reinforcement learning can be achieved using only a behaviour cloning policy, which means that its learned reward can be used to further improve a BC policy using additional online interactions. However, CSIL has only been demonstrated using simple feedforward network policies, so we discuss how such an imitation learning algorithm could be applied to more complex policy architectures, such as those including transformers and diffusion models.

1 Introduction

Many past successes in robot learning and reinforcement learning have consisted of policies initialized from human demonstrations and finetuned using online experience [23, 10, 11, 30]. This combination of behavioural cloning (BC) [25] and reinforcement learning (RL) [32] is a pragmatic and effective approach to learning complex tasks in a sample-efficient fashion. More recently, behavioural cloning has become popular due to the rapid development of rich generative models that can accurately capture the complex action distribution exhibited by humans solving complex robotic tasks [6, 4, 27, 37, 29, 22]. The advances in these models relative to RL algorithms and reward design methods suggests that behavioural cloning with generative models and human demonstration collection may be a more practical approach to learning effective robotic policies. However, behavioural cloning is limited by being a purely offline imitation learning method, which results in the requirement of a large offline dataset in lieu of learning from online experience. BC also suffers from covariate shift [28], where incorrect actions in unseen states accumulate errors and results in poor performance.

This work reviews the coherent soft imitation learning algorithm (CSIL) [36], which proposes an approach of improving a BC policy using on- or offline reinforcement learning. This is achieved through inverse RL (IRL) [20], by showing that the BC policy can be used to define a 'shaped' reward function. This learned shaped reward allows the agent to learn how to act in states outside of the demonstration distribution, and thus learn to overcome the covariate shift problem.

*For further details see joemwatson.github.io/csil and github.com/google-deepmind/csil.

NeurIPS 2023 Workshop on Robot Learning: Pretraining, Fine-Tuning, and Generalization with Large Scale Models, New Orleans, United States of America

The goal of this paper is to discuss how expressive BC policies can be improved with online experience. Section 2 reviews CSIL and Section 3 discusses CSIL's extension to more expressive policies.

2 Coherent soft imitation learning

We consider the entropy-regularized setting [24, 16, 7] with Markov decision process $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, \mu_0 \rangle$, with states $\boldsymbol{s} \in \mathcal{S}$, actions $\boldsymbol{a} \in \mathcal{A}$, dynamics $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$, reward $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, discount factor γ, and initial state distribution μ_0. The stationary distribution under policy π is μ_π. The RL objective

$$\max_q \mathbb{E}_{\boldsymbol{s}_{t+1} \sim \mathcal{P}(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t),\, \boldsymbol{a}_t \sim q(\cdot|\boldsymbol{s}_t),\, \boldsymbol{s}_0 \sim \mu_0(\cdot)} [\textstyle\sum_t \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t)] - \alpha\, \mathbb{E}_{\boldsymbol{s} \sim \mu_q(\cdot)} [\mathbb{D}_{\mathrm{KL}}[q(\boldsymbol{a} \mid \boldsymbol{s}) \,||\, p(\boldsymbol{a} \mid \boldsymbol{s})]],$$

yields a 'soft' value function $\mathcal{V}_\alpha$ where $\alpha \geq 0$. The soft Bellman equation for critic $\mathcal{Q}$ is

$$\begin{aligned} \mathcal{Q}(\boldsymbol{s}, \boldsymbol{a}) &= r(\boldsymbol{s}, \boldsymbol{a}) + \gamma\, \mathbb{E}_{\boldsymbol{s}' \sim \mathcal{P}(\cdot|\boldsymbol{s}, \boldsymbol{a})} [\mathcal{V}_\alpha(\boldsymbol{s}')], \quad \mathcal{V}_\alpha(\boldsymbol{s}) = \alpha \log \textstyle\int \exp(\mathcal{Q}(\boldsymbol{s}, \boldsymbol{a})/\alpha) p(\boldsymbol{a} \mid \boldsymbol{s})\, \mathrm{d}\boldsymbol{a}, \\ &\geq r(\boldsymbol{s}, \boldsymbol{a}) + \gamma\, \mathbb{E}_{\boldsymbol{a}' \sim q(\cdot|\boldsymbol{s}'),\, \boldsymbol{s}' \sim \mathcal{P}(\cdot|\boldsymbol{s}, \boldsymbol{a})} [\mathcal{Q}(\boldsymbol{s}', \boldsymbol{a}') + \alpha(\log q(\boldsymbol{a}' \mid \boldsymbol{s}') - \log p(\boldsymbol{a}' \mid \boldsymbol{s}'))], \end{aligned}$$

where the lower bound is easier to optimize when using the current policy q. The policy update is a closed-form pseudo-posterior [24, 1, 34] combining the policy prior and critic likelihood,

$$q_\alpha(\boldsymbol{a} \mid \boldsymbol{s}) \propto \exp((\mathcal{Q}(\boldsymbol{s}, \boldsymbol{a}) - \mathcal{V}_\alpha(\boldsymbol{s}))/\alpha)\, p(\boldsymbol{a} \mid \boldsymbol{s}). \tag{1}$$

To perform inverse RL, we invert the policy update in Equation 1 to obtain an expression of the critic, and plug this critic into the soft Bellman equation to obtain an expression for the reward [36],

$$\mathcal{Q}(\boldsymbol{s}, \boldsymbol{a}) = \alpha \log \frac{q(\boldsymbol{a} \mid \boldsymbol{s})}{p(\boldsymbol{a} \mid \boldsymbol{s})} + \mathcal{V}_\alpha(\boldsymbol{s}), \quad r(\boldsymbol{s}, \boldsymbol{a}) = \alpha \log \frac{q(\boldsymbol{a} \mid \boldsymbol{s})}{p(\boldsymbol{a} \mid \boldsymbol{s})} + \mathcal{V}_\alpha(\boldsymbol{s}) - \gamma\, \mathbb{E}[\mathcal{V}_\alpha(\boldsymbol{s}')]. \tag{2}$$

Applying reward shaping theory [17], the log policy ratio represents a shaped reward, sharing the same optimal policy. As the BC policy is optimal for this shaped reward, we call it a 'coherent'.

Definition 1. *The shaped 'coherent' reward and critic are derived from the log policy ratio introduced in Section 2, with value function $\mathcal{V}_\alpha(\boldsymbol{s})$ as the potential $\Psi(\boldsymbol{s})$. When policy $q_\alpha(\boldsymbol{a}|\boldsymbol{s})$ models the data $\mathcal{D}$ while matching its prior p otherwise, their density ratio should exhibit the following shaping*

$$\tilde{r}(\boldsymbol{s}, \boldsymbol{a}) = \alpha \log \frac{q_\alpha(\boldsymbol{a} \mid \boldsymbol{s})}{p(\boldsymbol{a} \mid \boldsymbol{s})} \begin{cases} \geq 0 \text{ if } \boldsymbol{s}, \boldsymbol{a} \in \mathcal{D}, \\ < 0 \text{ if } \boldsymbol{s} \in \mathcal{D}, \boldsymbol{a} \notin \mathcal{D}, \\ = 0 \text{ if } \boldsymbol{s} \notin \mathcal{D}, \forall \boldsymbol{a} \in \mathcal{A}. \end{cases}$$

In continuous setting, this shaping should be approximately captured by the policy, as shown by Figure 1.

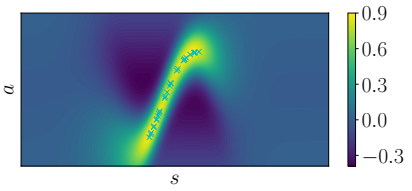

Figure 1: A coherent reward made using a stationary Gaussian process policy, extended out-of-distribution. The reward approximates the shaping described in Definition 1. The figure is taken from Watson et al. [36].

Algorithm 1 shows how CSIL is implemented with function approximation given a policy and critic loss $\mathcal{J}_\pi$ and $\mathcal{J}_\mathcal{Q}$. These two losses depend on the choice of soft policy iteration method.

Algorithm 1: Off-policy coherent soft imitation learning with function approximation

Data: Expert demonstrations $\mathcal{D}$, initial temperature α, refinement temperature, β,
parametric policy class $q_{\boldsymbol{\theta}}(\boldsymbol{a} \mid \boldsymbol{s})$, prior policy $p(\boldsymbol{a} \mid \boldsymbol{s})$, regression regularizer Ψ, total steps T
Result: $q_{\boldsymbol{\theta}_N}(\boldsymbol{a} \mid \boldsymbol{s})$, matching or improving the initial policy $q_{\boldsymbol{\theta}_1}(\boldsymbol{a} \mid \boldsymbol{s})$

```
// Pretrain coherent reward and critic
```

Train initial policy from demonstrations, $\boldsymbol{\theta}_1 = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{s}, \boldsymbol{a} \sim \mathcal{D}} [\log q_{\boldsymbol{\theta}}(\boldsymbol{a} \mid \boldsymbol{s}) - \Psi(\boldsymbol{\theta})]$;
Define fixed shaped coherent reward, $\tilde{r}_{\boldsymbol{\theta}_1}(\boldsymbol{s}, \boldsymbol{a}) = \alpha(\log q_{\boldsymbol{\theta}_1}(\boldsymbol{a} \mid \boldsymbol{s}) - \log p(\boldsymbol{a} \mid \boldsymbol{s}))$;
Pretrain critic with SARSA on $\mathcal{D}$ using $\tilde{r}_{\boldsymbol{\theta}_1}$.
for $t = 1 \to T$ **do**

```
// finetune policy with reinforcement learning
```

  Interact with environment $\boldsymbol{s}_{t+1} = \mathrm{Env}(\boldsymbol{s}_t, \boldsymbol{a}_t)$, $\boldsymbol{a}_t \sim q_{\boldsymbol{\theta}_t}(\cdot \mid \boldsymbol{s}_t)$, store in replay buffer $\mathcal{B}$;
  Optimize reward and critic using $\mathcal{J}_\mathcal{Q}(\boldsymbol{\phi})$ and $\mathcal{J}_r(\boldsymbol{\theta})$ and $\mathcal{J}_\pi(\boldsymbol{\theta})$ on minibatch from $\mathcal{B}$;
end

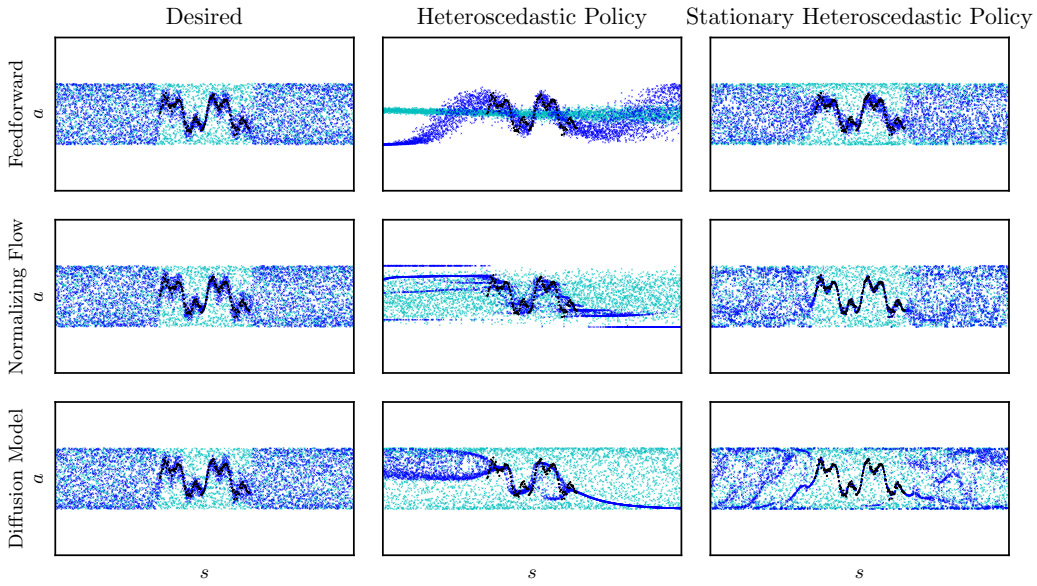

Figure 2: For the coherent reward to be effective, the stochastic policy should return to the prior distribution outside of the data distribution (left). The typical heteroscedastic parametric policies have undefined out-of-distribution behaviour and typically collapse to the action limits due to the network extrapolation and tanh transformation (middle). By approximating stationary Gaussian processes, we can design policies that exhibit the desired behavior with minimal network modifications (right). This figure extends the one in Watson et al. [36] to more expressive policies, diffusion models and normalizing flows. Interestingly, not only do the standard architectures result in the same undefined out-of-distribution behaviour, but using a stationary feature space in the right way acts to mitigate this effect. The figure shows samples from the initial policy ● and trained policy ●.

Soft actor critic (SAC) [7] uses the squared Bellman error for $\mathcal{J}_\mathcal{Q}$ and the reperameterization trick for the policy update, derived from the I-projection $\min_{\boldsymbol{\theta}} \mathbb{D}_{\text{KL}}[q_{\boldsymbol{\theta}}(\boldsymbol{a} \mid \boldsymbol{s}) \mid\mid q_\alpha(\boldsymbol{a} \mid \boldsymbol{s})]$,

$$\mathcal{J}_\pi(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{a}\sim q_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}),\, \boldsymbol{s}\sim\mathcal{B}}\left[\mathcal{Q}(\boldsymbol{s},\boldsymbol{a}) - \alpha\left(\log q_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{s}) - \log p(\cdot \mid \boldsymbol{s})\right)\right]. \tag{3}$$

One specific requirement for CSIL that is not required of SAC is 'stationary' policy architectures. This property ensures that q approximates a constant prior $p(\boldsymbol{a} \mid \boldsymbol{s}) = p(\boldsymbol{a})$ at initialisation and retains this property after training in out-of-distribution regions. This stationary property can be enforced through an augmented objective [35], but can be achieved using a minor network architecture modification [15]. For more details, see Watson et al. [36]. Figure 2 shows how this stationarity property can be achieved, and in this work we demonstrated this for normalizing flow and diffusion policies. The implementation details are in Appendix B.

Due to the coherence property of its shaped reward, CSIL is able to perform imitation learning that not only improve the BC policy out-of-distribution, but also does not result in unlearning the initial policy, which occurs in non-coherent imitation learning [19, 36]. Figure 3 shows CSIL's performance on the `robomimic` tasks [13], which consist of learning manipulation tasks from human demonstrations. Figure 4 in the Appendix shows learning over environment steps, demonstrating the sample efficiency of CSIL and how it leverage the BC policy initialization without unlearning. Impressive importance in `robomimic` was also demonstration by BC policies that use recurrent architectures and diffusion models over action sequences Chi et al. [4]. These policies were also capable from learning from sub-optimal human demonstrations, which may exhibie multi-modality and pauses in motion. The next section discusses how CSIL may be applied to more complex policies such as these, in order to leverage the capabilities of both expressive BC and online IRL.

3 The considerations for online imitation with expressive policies

CSIL was evaluated using a stationary variation of the SAC policy, which is a feedforward network parameterizing a Gaussian distribution which is then clamped through a tanh transformation in order to satisfy the action limits. More advanced policies take the form $[\boldsymbol{a}_t, \dots] \sim \pi(\cdot \mid \boldsymbol{s}_t, \dots)$,

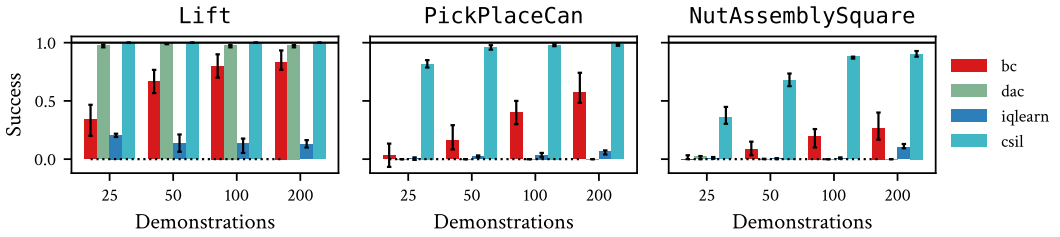

Figure 3: Average success rate over 50 evaluations for online imitation learning for `robomimic` tasks. Uncertainty intervals depict quartiles over 10 seeds. To assess convergence across seeds, performance is chosen using the highest 25th percentile of the averaged success during learning. These results are from Watson et al. [36], using simple Tanh Gaussian policies.

conditioning on a sequence of observations to sample a sequence of actions from a more expressive conditional distribution π. We now review these properties in the context of online imitation learning.

Generative models for actions. Many RL algorithms require tractable loglikelihood in order to update the policy and compute the policy's entropy. Normalizing flows (NF) [21] are one class of generative model with tractable loglikelihood, and in fact the tanh policy used by SAC can be seen as a simple normalizing flow. Diffusion models [9] do not have available loglikelihoods by default, as the score function predicts noisy perturbations of the data. FlowMatching [12] and ScoreFlow [31] both demonstrate diffusion-like models with available log-likelihoods by connection diffusion models to continuous normalizing flows. Stationary conditional generative models have, to the best of our knowledge, not been investigated. Figure 2 and Appendix B show how this property is approximated using the architectural modifications proposed by Meronen et al. [15] and Watson et al. [36].

Recurrent architectures. Recurrent policies have previously been investigated in reinforcement learning, e.g. [8], and can be used with similarly recurrent critic architecture, although careful implemenation is required to stabilize training and ensure good performance [18].

Action sequence prediction. To ensure temporal consistency with the demonstrations, policies can sample a sequence of H actions every H timesteps that are executed in an open-loop fashion, i.e. $[\boldsymbol{a}_t, \dots \boldsymbol{a}_{t+H}] \sim \pi(\cdot \mid \boldsymbol{s}_t, \dots)$ [4, 37]. In order to do policy evaluation with this approach, the critic would have to consider the extended action space, i.e. $\mathcal{Q}(\boldsymbol{s}_t, \boldsymbol{a}_t, \dots, \boldsymbol{a}_{t+H})$, and consider the future value at $\mathcal{V}_\alpha(\boldsymbol{s}_{t+H+1})$. Given that accurate policy evaluation with function approximation remains a central challenge in RL, it is an open question how current methods scale to an extended action space.

Prior related work. NF policies have been incorporated into SAC due to the availability of the loglikelihoods and reparameterized gradients [14]. Imitation learning with expressive policies has predominantly used BC as discussed above. Zhao et al. [37] used a conditional variational autoencoder as the generative distribution for a recurrent transformer-based BC policy. Expressive policies and critics have mainly been investigated in reinforcement learning in the offline setting. Wang et al. [33] combined BC with the critic guidance from Equation 3 to train a diffusion policy. Transformer-based critics have also been applied in the offline setting, but only by discretizing the continuous action space [2]. The decision transformer [3] treats RL as a sequence modelling and compute actions by conditioning on an estimate of the largest possible return.

4 Discussion

We have discussed how the recent trend of recurrent and generative policy architectures can finetuned online in a sample-efficient manner using the CSIL algorithm. However, for the CSIL algorithm to be applicable, two design design decisions are required: tractable log-likelihood computation and a stationarity. We have demonstrated how the stationary property can be incorporated into normalizing flows and diffusion models. Tractable log-likelihood computation is available from a subset of generative models, namely normalizing flows and flow matching. It remains to be seen if the additional design decisions described in Section 3, recurrent critics and extending the action space, result in effective policy evaluation that facilitates online improvement in a scalable fashion.

Acknowledgments

This paper is primarily a review of Watson et al. [36], which was undertaken at Google Deepmind in collaboration with Sandy Huang and Nicolas Heess during an internship. The motivation to extend CSIL to richer policies was inspired by conversations with Moritz Reuss. Joe Watson acknowledges the grant "Einrichtung eines Labors des Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI) an der Technischen Universität Darmstadt" of the Hessian Ministry of Science and Art.

References

[1] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a Posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

[2] Y. Chebotar, K. Hausman, F. Xia, Y. Lu, A. Irpan, A. Kumar, T. Yu, A. Herzog, K. Pertsch, K. Gopalakrishnan, et al. Q-Transformer: Scalable offline reinforcement learning via autoregressive Q-functions. In *Conference on Robot Learning*, 2023.

[3] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 2021.

[4] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

[5] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. Neural spline flows. *Advances in Neural Information Processing Systems*, 2019.

[6] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, 2022.

[7] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

[8] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver. Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*, 2015.

[9] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 2020.

[10] J. Kober and J. Peters. Policy search for motor primitives in robotics. *Advances in Neural Information Processing Systems*, 21, 2008.

[11] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 2016.

[12] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2022.

[13] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning*, 2021.

[14] B. Mazoure, T. Doan, A. Durand, J. Pineau, and R. D. Hjelm. Leveraging exploration in off-policy algorithms via normalizing flows. In *Conference on Robot Learning*, 2020.

[15] L. Meronen, M. Trapp, and A. Solin. Periodic activation functions induce stationarity. In *Advances in Neural Information Processing Systems*, 2021.

[16] G. Neu, A. Jonsson, and V. Gómez. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.

[17] A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, 2000.

[18] T. Ni, B. Eysenbach, and R. Salakhutdinov. Recurrent model-free RL can be a strong baseline for many POMDPs. In *International Conference on Machine Learning*. PMLR, 2022.

[19] M. Orsini, A. Raichuk, L. Hussenot, D. Vincent, R. Dadashi, S. Girgin, M. Geist, O. Bachem, O. Pietquin, and M. Andrychowicz. What matters for adversarial imitation learning? In *Advances in Neural Information Processing Systems*, 2021.

[20] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 2018.

[21] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 2021.

[22] T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann, and S. Devlin. Imitating human behaviour with diffusion models. In *International Conference on Learning Representations*, 2023.

[23] J. Peters and S. Schaal. Policy gradient methods for robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[24] J. Peters, K. Mülling, and Y. Altün. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2010.

[25] D. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 1991.

[26] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *Advances in Neural Information Processing systems*, 20, 2007.

[27] M. Reuss, M. Li, X. Jia, and R. Lioutikov. Goal conditioned imitation learning using score-based diffusion policies. In *Robotics: Science and Systems*, 2023.

[28] S. Ross and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[29] L. X. Shi, A. Sharma, T. Z. Zhao, and C. Finn. Waypoint-based imitation learning for robotic manipulation. *arXiv preprint arXiv:2307.14326*, 2023.

[30] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

[31] Y. Song, C. Durkan, I. Murray, and S. Ermon. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 2021.

[32] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[33] Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations*, 2022.

[34] J. Watson and J. Peters. Inferring smooth control: Monte Carlo posterior policy iteration with Gaussian processes. In *Conference on Robot Learning*, 2022.

[35] J. Watson, J. A. Lin, P. Klink, and J. Peters. Neural linear models with functional Gaussian process priors. In *Symposium on Advances in Approximate Bayesian Inference*, 2020.

[36] J. Watson, S. H. Huang, and N. Heess. Coherent soft imitation learning. *Advances in Neural Information Processing Systems*, 2023.

[37] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023.

A Additional experimental results

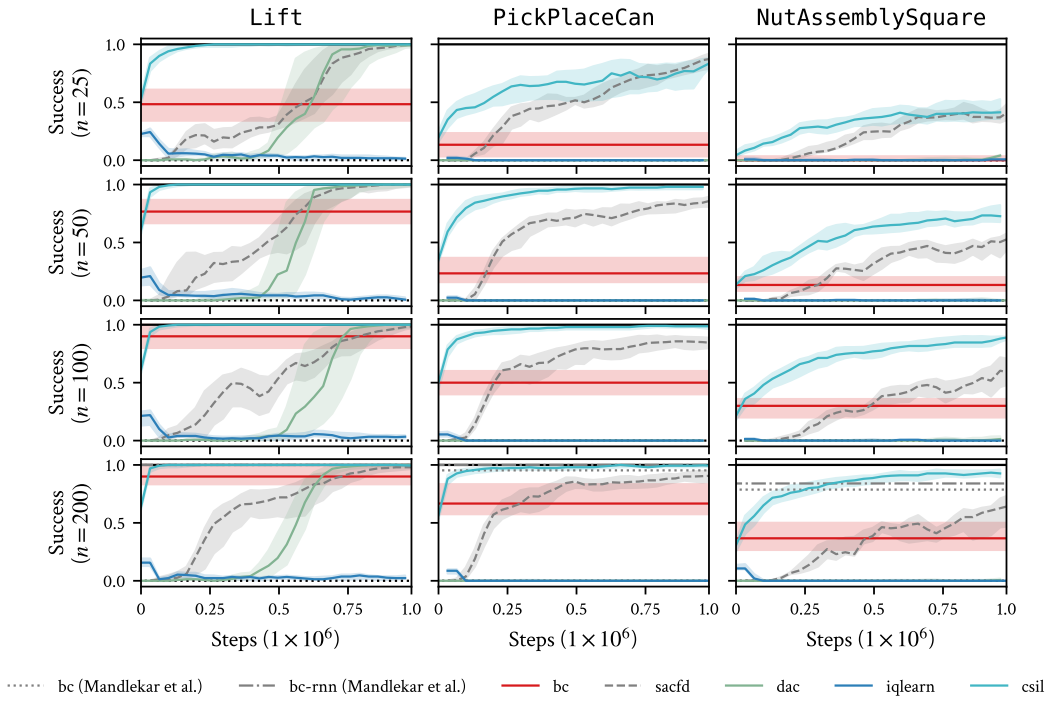

Figure 4: Average success rate over 50 evaluations for online imitation learning for `robomimic` tasks. Uncertainty intervals depict quartiles over 10 seeds. The BC policy is trained on the demonstration data. n refers to demonstration trajectories. Taken from Watson et al. [36].

B Implementation details

The nominal SAC policy defines a latent stochastic policy using a feedforward network with two heads, where one predicts the mean and other parameterizes the predictive variance by adding a transform that ensures positivity. The latent action is then passed through a tanh to ensure the action constraints, under the assumption the action space in normalized such that $a \in [-1, 1]$. The policy can be expressed as $\boldsymbol{a} = \tanh(\boldsymbol{z}(\boldsymbol{s}))$, where $\boldsymbol{\mu}_{\boldsymbol{z}}(\boldsymbol{s}) = \boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{s})$, where $\boldsymbol{\phi}$ is the shared network feature space that defines both the mean and variance.

The stationary HETSTAT policy in Figure 2 is achieved by defining a stationary parametric Gaussian process (GP) using a specific feature space ϕ. The GP is alternatively defined as $\boldsymbol{z}(\boldsymbol{s}) = \boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{s})$ with Gaussian weights $\boldsymbol{W}$. The weights are factorized row-wise $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{d_a}]^\top$, $\boldsymbol{w}_i = \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ to define a GP with independent actions. Using change-of-variables like SAC [7], the policy is expressed per-action as

$$q(a_i \mid \boldsymbol{s}) = \mathcal{N}\left(z_i; \boldsymbol{\mu}_i^\top \boldsymbol{\phi}(\boldsymbol{s}),\ \boldsymbol{\phi}(\boldsymbol{s})^\top \boldsymbol{\Sigma}_i\, \boldsymbol{\phi}(\boldsymbol{s})\right) \cdot \left| \det\left(\frac{\mathrm{d}a_i}{\mathrm{d}z_i}\right) \right|^{-1}, \quad \boldsymbol{\phi}(\boldsymbol{s}) = f_{\text{per}}(\tilde{\boldsymbol{W}}\boldsymbol{\phi}_{\text{mlp}}(\boldsymbol{s})). \quad (4)$$

The stationary features $\boldsymbol{\phi}$ are achieved using periodic activation f_{per} in the spirit of the random Fourier feature approximation of the squared exponential stationary covariance function [26], but can be non-sinusoidal function, e.g. triangle waves or periodic ReLUs [15]. The pre-periodic weight distribution $\tilde{W}_{ij} \sim \mathcal{N}(0, 1)$, which is another requirement of the stationary covariance function approximation.

B.1 Normalizing flows

Normalizing flow policies consist of a latent heteroscedastic policy where $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{s}))$ and action sample $\boldsymbol{a} = \varphi_{\boldsymbol{p}}(\boldsymbol{s})$ where $\varphi_{\boldsymbol{p}}$ is a parametric bijective function. For the normalizing flow

policy, $\varphi_{\boldsymbol{p}}$ was made up of a tanh layer followed by five parametric rational quadratic spline flows [5] defined between $[-1, 1]$, as we assume actions are normalized to this range. We chose spline flows because they are expressive and invertible. The bijection parameters $\boldsymbol{p}$ are learned using a function approximator. For parameter efficiency we share a torso with the latent policy so $\boldsymbol{p} = \boldsymbol{P}\boldsymbol{\phi}(\boldsymbol{s})$.

For the stationary approximation in Figure 2, we use the stationary parametric Gaussian process from Watson et al. [36] for the latent policy, and define the spline network using the stationary features. For the splines, zero-valued parameters result in the identity mapping, so a zero-mean prior for that network results in the HETSTAT policy from Watson et al. [36].

B.2 Diffusion models

Our conditional diffusion models are defined by a parametric score function $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{a}, t)$ where action samples $\boldsymbol{a}_0$ are obtained through repeating $\boldsymbol{a}_{t-\tau} = \alpha(t)\,\boldsymbol{a}_t - \beta(t)\,\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{a}_t, t) + \sigma(t)\,\boldsymbol{v}_t$ where $t \in [0, 1]$, discretization $\tau = 1/N$ for N sampling steps, $\boldsymbol{v}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $\boldsymbol{a}_1 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Schedules α, β and σ are designed in Ho et al. [9]. The objective for learning the score

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{a}\sim\mathcal{D},\, t\sim\mathcal{U}(0,1),\, \boldsymbol{v}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}[\omega(t)|\boldsymbol{v} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{s}, \gamma(t)\boldsymbol{a} + \kappa(t)\boldsymbol{v}, t)|^2], \tag{5}$$

as shown in Equation 12 of Ho et al. [9] where temporal weightings ω, γ and κ are defined.

For the stationary approximation in Figure 2, we simply parameterize $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ as the stationary mean function which is zero. If $\boldsymbol{\epsilon}(\boldsymbol{s}, \boldsymbol{a}, t) = 0 \;\forall\; \boldsymbol{s} \in \mathcal{S}, \boldsymbol{a} \in \mathcal{A}, t \in [0, 1]$, then the predictive samples $\boldsymbol{a}_0 \sim \mathcal{N}(\mathbf{0}, \sigma_0 \boldsymbol{I})$ where σ_0 depends on the noise schedules $\alpha(t)$ and $\sigma(t)$.