IMPLICIT TWO-TOWER POLICIES

Yunfan Zhao[*1], **Qingkai**[*2], **Krzysztof Choromanski**[*3,4], **Deepali Jain**[3], **Vikas Sindhwani**[3]

ABSTRACT

We present a new class of structured reinforcement learning policy-architectures, *Implicit Two-Tower (ITT)* policies, where the actions are chosen based on the attention scores of their learnable latent representations with those of the input states. By explicitly disentangling action from state processing in the policy stack, our approach allows training and inference on resource constrained devices (e.g. low memory, low wall-clock time allowance). Our architectures are compatible with both discrete and continuous action spaces. By conducting tests on 15 environments from OpenAI Gym and DeepMind Control Suite, we show that ITT-architectures outperform their vanilla unstructured implicit counterparts as well as commonly used explicit policies. We complement our analysis by showing that ITT, compatible with techniques such as hashing and lazy tower updates, is particular well-suited for low resource settings.

1 INTRODUCTION & RELATED WORK

We consider the problem of training a policy $\pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$, parameterized by learnable $\theta \in \mathbb{R}^D$ for a reinforcement learning (RL) agent (Sutton & Barto (1998); Sutton (1998); Bartlett (2002); Singi et al. (2023); Zhou et al. (2022); He et al. (2023a); Huang & Wang (2020)). The policy is a potentially stochastic mapping from the state-space ($\mathcal{S}$) to the action-space ($\mathcal{A}$), either continuous or discrete. The objective is to maximize the expected total reward R defined as a possibly discounted sum of the partial rewards $r_i(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}_{i+1})$ for the transition from $\mathbf{s}_i \in \mathcal{S}$ to $\mathbf{s}_{i+1} \in \mathcal{S}$ via $\mathbf{a}_i \in \mathcal{A}$. The transition function: $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ as well as the partial reward function: $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ (both potentially stochastic) are defined by the environment. Hence, expected total rewards are computed over random state transitions and action choices. We call the sequence $(\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, ..., \mathbf{s}_T)$ of states visited by the agent intertwined with the actions proposed by π_θ, the *rollout* of an agent.

The most common way of encoding policy mapping π_θ is via a neural network taking states as inputs and explicitly outputting as the activations of the last layer proposed actions or the distributions over actions to sample from (for stochastic policies). We call such policies *explicit*. While explicit policies were successfully applied in several RL scenarios: learning directly from pixels, hierarchical learning, robot locomotion and more (Schulman et al. (2017); Salimans et al. (2017); Ha & Schmidhuber (2018); Choromanski et al. (2018); Yu et al. (2021); Jain et al. (2020); Huang & Wang (2022)), recent evidence shows that the expressiveness of the policy-architecture can be improved if the explicit model is substituted by an *implicit* one.

The implicit model (Haarnoja et al. (2017); Florence et al. (2021); Du et al. (2019)) operates by learning a function $E_\theta : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ taking as an input a state-action pair and outputting a scalar value that can be interpreted as an energy Xie et al. (2016); Xu et al. (2022). The optimal action $\mathbf{a}^*(\mathbf{s})$ for a given state $\mathbf{s}$ is chosen as a solution to the following energy-minimization problem:

$$\mathbf{a}^*(\mathbf{s}) = \pi_\theta(\mathbf{s}) = \mathrm{argmin}_{\mathbf{a} \in \mathcal{A}} E_\theta(\mathbf{s}, \mathbf{a}). \tag{1}$$

Implicit models were recently demonstrated to provide strong performance in the behavioral cloning (BC) setting (Florence et al. (2021)), outperforming their regular explicit counterparts (e.g. mean squared error and mixture density BC policies), also for high-dimensional action-spaces and image inputs. Interestingly, robots with deployed implicit policies were shown to learn sophisticated behaviours on various manipulation tasks requiring very high precision (Florence et al. (2021)).

New results, showing that the implicit mappings from states to actions given by Eq. 1 are capable of modeling multi-valued and even discontinuous functions with arbitrary precision (see: Theorem

[*]*Equal contribution [1] Harvard University [2] New York University [3] Google [4] Columbia University

1 and 2 in Florence et al. (2021), and Bianchini et al. (2021)) with continuous energy-functions E modeled by regular neural networks, shed light on that phenomenon. Thus adding the argmin-operator provides a gateway to extend universal approximation results of regular neural networks to larger classes of functions, enabling us to approximate with our neural network models classes of functions that cannot be approximated with regular neural networks (e.g. discontinuous functions).

In the standard implementation of the implicit policies (see for instance: Florence et al. (2021)), that we will refer to as the *implicit one-tower* (IOT) policies, the state and action feature vectors are concatenated and such an input is given to the energy-network. While seemingly natural, this approach has one crucial weakness - it is prohibitively expensive for large action-spaces. It requires solving nontrivial optimization argmin-problem at every state-to-state transition without opportunity to at least partially reuse computations conducted in the past. Indeed, even if the same actions are probed for different transitions (e.g. when the action-space has moderate cardinality or sampling heuristics are applied and the same sets of action samples are applied across different state-transitions) those actions are concatenated with different states leading to different inputs to the energy-function for different transitions. That is why in practice implicit policies apply sampling techniques with relatively few samples, affecting the approximation quality of the original argmin-problem.

To address this key limitation, we introduce a new class of implicit policies architectures, called *implicit two-towers* (ITTs) (Fig. 1), where action processing is explicitly disentangled from state processing via the architectural design. The ITT-architecture consists of two towers mapping states and actions to the same d-dimensional latent space $\mathcal{L}$. The negated energy $-E$ is then defined as a relatively simple kernel $\mathrm{K} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ acting on the state/action latent representations (e.g. dot-product or softmax kernel). In Appendix A, we discuss several classes of methods related to the implicit policies. **Our Main Contributions Are:**

- We propose a new class of structured reinforcement learning architecture, implicit two-tower policies. We demonstrate the new architecture achieves substantial computational savings and allows training and inference on resource constrained devices, while providing stronger performance than existing implicit policies and their explicit counterparts.
- We demonstrate that our ITT architecture is compatible with fast maximum inner product search algorithms, achieving additional computational savings and is particularly well-suited for low resource settings (low memory, low wall-clock time allowance).
- By disentangling action and state processing, ITTs allow for state and action towers to be updated at different rates and makes it possible to reuse computations conducted for a fixed set of actions. Thus, we reduce memory needs achieve further computational gains.

2 IMPLICIT TWO-TOWER POLICIES

2.1 PRELIMINARIES

As described in Section 1, we focus in this paper on the implicit policies $\pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$ from the state-space $\mathcal{S} \subseteq \mathbb{R}^s$ to the action-space $\mathcal{A} \subseteq \mathbb{R}^a$, given as follows for the learnable $\theta \in \mathbb{R}^D$:

$$\pi_\theta(\mathbf{s}) = \mathrm{argmin}_{\mathbf{a} \in \mathcal{A}} E_\theta(\mathbf{s}, \mathbf{a}) \tag{2}$$

Here $E_\theta : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the *energy-function*, usually encoded by the neural network. In the standard implicit-policy approach, the *one-tower* model (IOT), the input to this neural network is the concatenated state-action vector: $\mathrm{input} = [\mathbf{s}, \mathbf{a}]$. Solving optimization problem from Eq. 2 directly is usually prohibitively expensive. Thus instead sampling strategies are often used. For the selected set $\mathcal{A}^* = \{\mathbf{a}_1, ..., \mathbf{a}_N\}$ of sampled actions (usually uniformly at random from $\mathcal{A}$), the algorithm approximates $\pi_\theta(\mathbf{s})$ as: $\widehat{\pi}_\theta(\mathbf{s}) = \mathrm{argmin}_{\mathbf{a} \in \mathcal{A}^*} E_\theta(\mathbf{s}, \mathbf{a})$ or applies derivative-free-optimization heuristics (see: Florence et al. (2021)). Alternatively, the task is relaxed and instead of solving the original argmin-problem, the action $\mathbf{a} \in \mathcal{A}^*$ is sampled from the following softmax-distribution defined on $\mathcal{A}^*$ (the relaxation allows backpropagating through the action-selection modules):

$$\mathbb{P}[\widehat{\pi_\theta}(\mathbf{s}) = \mathbf{a}_i] = \frac{\exp(-E_\theta(\mathbf{s}, \mathbf{a}_i))}{\sum_{\mathbf{a} \in \mathcal{A}^*} \exp(-E_\theta(\mathbf{s}, \mathbf{a}))} \tag{3}$$

All the aforementioned approaches are inherently linear in the number of sampled actions. Furthermore, sampling is usually conducted at every state-transition. Thus in practice, for computational efficiency, a small number of samples needs to be used.

2.2 TWO TOWERS

We propose the implicit two-tower (ITT) model, where the energy is defined as:

$$E_\theta(\mathbf{s}, \mathbf{a}) = -\mathrm{K}(l_{\mathcal{S}}^{\theta_1}(\mathbf{s}), l_{\mathcal{A}}^{\theta_2}(\mathbf{a})) \tag{4}$$

for the state-tower mapping: $l_{\mathcal{S}}^{\theta_1}(\mathbf{s}) : \mathbb{R}^s \rightarrow \mathcal{L} \subseteq \mathbb{R}^d$ and action-tower mapping: $l_{\mathcal{A}}^{\theta_2}(\mathbf{a}) : \mathbb{R}^a \rightarrow \mathcal{L} \subseteq \mathbb{R}^d$, parameterized by $\theta_1 \in \mathbb{R}^{D_1}$ and $\theta_2 \in \mathbb{R}^{D_2}$ respectively (usually encoded by two neural networks) as well as a fixed kernel function $\mathrm{K} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$. Here $\mathcal{L}$ stands for the common latent space for states and actions. As in the case of regular implicit policies, action selection is conducted by solving the argmin-problem or its softmax-sampling relaxation.

A particularly prominent class of kernels that can be applied are those that are increasing functions of the dot-products of their inputs, i.e. $\mathrm{K}(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}^\top \mathbf{y})$ for some $f : \mathbb{R} \rightarrow \mathbb{R}$. Those include dot-product kernel, where f is an identity function as well as the softmax-kernel, where $f(z) \stackrel{\mathrm{def}}{=} \exp(z)$. For those kernels the corresponding argmax-problems are trivially equivalent and reduce to the maximum-inner-product (MIP) search Shrivastava & Li (2014); Neyshabur & Srebro (2015); Sundaram et al. (2013), but the softmax-distributions differ. For a fixed sampled set of actions $\mathcal{A}^* = \{\mathbf{a}_1, ..., \mathbf{a}_N\}$, the MIP problem can be solved particularly efficiently as follows:

$$\widehat{\pi}_{\theta_1,\theta_2}(\mathbf{s}) = \mathbf{a}_{\mathrm{argmax}\left(l_{\mathcal{A}}^{\theta_2}(\mathcal{A}^*) l_{\mathcal{S}}^{\theta_1}(\mathbf{s})\right)}, \tag{5}$$

where the i^{th} rows of the matrix $l_{\mathcal{A}}^{\theta_2}(\mathcal{A}^*) \in \mathbb{R}^{N \times d}$ is given as $l_{\mathcal{A}}^{\theta_2}(\mathbf{a}_i)$. Thus brute-force computation of the action for the given state between two consecutive updates of the action-tower (or: the set of sampled actions) takes time: $O(Nd + T_{\mathcal{S}})$, where $T_{\mathcal{S}}$ is the time needed to compute latent representation $l_{\mathcal{S}}^{\theta_1}(\mathbf{s})$ of $\mathbf{s}$.

ITTs can in particular apply a rich set of techniques for solving the maximum inner product (MIP) problem Pham (2021); Shrivastava & Li (2015a;b); Pham (2020), such as LSH-hashing, as well as algorithms conducting sub-linear softmax-sampling via linearization of the softmax kernel with random feature trees Choromanski et al. (2021a); Rawat et al. (2019). We discuss details on adapting MIP techniques in Appendix B. Interestingly, they also produce policies obtaining larger rewards as compared to their IOT and explicit counterparts, as we demonstrate in Sec. 4 and in Appendix C on 15 environments taken from OpenAI Gym and DeepMind Control Suite.

3 ES-OPTIMIZATION SETUP

We decided to train parameters of our ITT-architectures with the class of Evolutionary Search (ES, Blackbox) methods (Salimans et al. (2017); Choromanski et al. (2018); Mania et al. (2018); Choromanski et al. (2019)). Even though ITTs are agnostic to the particular training algorithm, ES-methods constituted a particularly attractive option. They enabled us to benchmark implicit policies in the on-policy setting. Furthermore, as simple conceptually and very efficient, they let us focus on the architectural aspects rather than tedious hyperparameter-tuning. Finally, they work very well also with non-differentiable or even non-continuous objectives, fitting well the combinatorial-flavor of the energy optimization problem formulation in ITTs.

Let $\theta = (\theta_1^\top, \theta_2^\top)^\top \in \mathbb{R}^D$, where $D = D_1 + D_2$ and $\theta_i \in \mathbb{R}^{D_i}$ for $i = 1, 2$. We are allowed to query an objective $F(\cdot)$ that measures the expected discount cumulative reward of policy parameters $\theta \in \mathbb{R}^D$. The objective is commonly used in the RL literature and can be evaluated by running a trajectory with θ Choromanski et al. (2018); Sutton & Barto (2018); Dou et al. (2022); Zhou et al. (2023); He et al. (2023c); Huang & Wang (2023). We conducted gradient-based optimization with the antithetic ES-gradient estimator applying orthogonal samples (see: Choromanski et al. (2018)), defined as follows:

$$\widehat{\nabla}_M^{\mathrm{AT,\,ort}} F_\sigma(\theta) \coloneqq \frac{1}{2\sigma M} \sum_{i=1}^{M} F^{AT(i)} \tag{6}$$

$$\text{where } \; F^{AT(i)} \coloneqq F\left(\theta + \sigma\varepsilon_i\right)\varepsilon_i - F\left(\theta - \sigma\varepsilon_i\right)\varepsilon_i$$

for the hyper-parameter $\sigma > 0$ and where $(\varepsilon_i)_{i=1}^M$ have marginal distribution $\mathcal{N}(\mathbf{0}, I_D)$, and $(\varepsilon_i)_{i=1}^M$ are conditioned to be pairwise-orthogonal, and M is the number of perturbations we choose. Such an

ensemble of samples can always be constructed since in all our experiments we have: $M \leq D$. To be more specific, for the DMCS environments we used: $M = 500$ and for all other: $M = D$. At each training epoch, we were updating θ as

$$\theta_{k+1} = \theta_k + \eta \widehat{\nabla}_M^{\mathrm{AT,\,ort}} F_\sigma(\theta), \tag{7}$$

where the learning rate $\eta = 0.01$ is fixed throughout the experiments. We present in Appendix F **Theoretical Analysis** of the antithetic estimator , to shed light on its effectiveness (see Theorem F.4 and Lemma F.6).

4 EXPERIMENTS

For the $\mathrm{OpenAI\ Gym}$ environments, at each transition we were sampling a set of actions $\mathcal{A}^* = \{\mathbf{a}_1, ..., \mathbf{a}_N\}$ (see Equation 7), and choosing an action according to Equation 5. For the DMCS environments, $\mathcal{A}^*$ was sampled at each iteration of the ES-optimization (independently for different ES-workers). For the environments with discrete actions, such as MountainCar-v0, we choose $\mathcal{A}^* = \mathcal{A}$. For the environments with continuous actions, we set $N = 1000$ for the $\mathrm{OpenAI\ Gym}$ environments and $N = 10000$ for the DMCS environments.

In Table 1, we present final average scores together with their correesponding standard deviations (over $s = 10$ different seeeds) for all 15 environments tested in the paper. In Appendix C, we provide plots showing training curves. ITTs provide the best policies on 12 out of 15 tasks in terms of the final average score and is second best on the remaining three.

Table 1: We present final average scores over $s = 10$ random seeds together with their std. The best architecture is in bold font and the second best is underscored.

Environment	ITT	IOT	Explicit
Swimmer-v2	$\underline{344.13} \pm 5.18$	75.54 ± 88.05	**347.67** ± 5.74
LunarLanderContinuous-v2	**157.38** ± 71.81	-72.72 ± 26.98	$\underline{62.85} \pm 176.36$
Hopper-v2	**2670.37** ± 225.53	1036.52 ± 29.16	$\underline{1060.89} \pm 40.71$
HalfCheetah-v2	**2866.02** ± 416.81	$\underline{2696.29} \pm 274.51$	1845.64 ± 441.96
Walker2d-v2	$\underline{1897.87} \pm 498.86$	**2909.85** ± 621.45	1346.93 ± 906.33
CartPole-v1	**500.00** ± 0.00	**500.00** ± 0.00	**500.00** ± 0.00
MountainCar-v0	**-133.50** ± 27.10	-200.00 ± 0.00	$\underline{-143.40} \pm 37.12$
Acrobot-v1	$\underline{-82.60} \pm 11.93$	**-82.00** ± 8.54	-92.10 ± 21.41
MountainCarContinuous-v0	**89.17** ± 8.97	25.96 ± 53.61	$\underline{52.34} \pm 42.74$
InvertedPendulumBulletEnv-v0	**1000.00** ± 0.00	27.10 ± 12.70	**1000.00** ± 0.00
DMCS:Swimmer6	**988.26** ± 10.11	$\underline{984.88} \pm 28.05$	767.44 ± 120.22
DMCS:Swimmer15	**995.81** ± 7.02	$\underline{990.68} \pm 0.001$	935.99 ± 48.69
DMCS:FishSwim	**652.21** ± 152.07	331.73 ± 8.42	$\underline{470.66} \pm 0.004$

Wall-clock Time. Table 2 shows that for environments with lower dimensional action space (with the corresponding smaller sizes of policy-architectures) all three architectures perform similarly speed-wise. However for more complicated environments ITTs train much faster than IOTs (**26%** training time reduction for Hopper-v2, **45%** for HalfCheetah-v2 and **39%** for Walker2d-v2). We provide further analysis on computational savings in Appendix C.

Table 2: Comparison of wall-clock times (in hours on 24 CPUs) for different policy-architectures. "LunarLanderC-v2“ here stands for LunarLanderContinuous-v2

Environment	ITT	IOT	Explicit	# iter
Swimmer-v2	0.17	0.18	0.11	500
LunarLanderC-v2	0.06	0.06	0.08	500
Hopper-v2	2.49	3.36	2.42	4000
HalfCheetah-v2	12.94	23.61	9.95	4000
Walker2d-v2	16.88	27.49	13.41	4000

ITT with Random Feature Trees (ITT-RFT). Random feature mechanisms are shown to provide substantial computational gains with little sacrifice in performanceChoromanski et al. (2023) (see

Appendix B for details on how to use random feature trees with our ITT). Table 3 shows ITT-RFT substantially reduced the amount of samples required to reach given reward thresholds.

Table 3: We present the **number of timesteps** needed to reach given reward thresholds set by previous works (Salimans et al., 2017; Schulman et al., 2015). The results were averaged over 6 random seeds. "ES" stands for the results of explicit policies provided by (Salimans et al., 2017). "ITT-RFT timesteps" stands for ITT with random feature tree.

Environment	Reward threshold	ITT	ITT-RFT	ES	TRPO
Swimmer-v1	128.25	1.07e+06	0.82e+06	1.39e+06	4.59e+06
Hopper-v1	877.45	0.69e+05	0.67e+05	3.83e+05	7.29e+05
Walker2d-v1	957.68	3.16e+05	4.84e+05	6.43e+05	1.55e+06

ITT with Signed Random Projections (ITT-SRP). ITTs can be used with popular signed random projection techniques to further reduce sample complexity and wall-clock time. (see Appendix B for details). We take $N = 2^c$ action samples per timestep. We observe that when the number of samples required gets large, ITT-SRP trains much faster than IOTs (for HalfCheetah-v1 **92%** training time reduction for $c = 14$, **89%** for $c = 13$, **81%** for $c = 12$). Furthermore, IOT wall-clock time grows exponentially in the parameter c, and ITT-SRP wall-clock time only grows linearly in the parameter c. We also conducted paired t-tests showing that the wall-clock time savings of ITT-SRP is statistically significant at a 95% confidence level.

Above results on ITT-RFT and ITT-SRP showcase that ITT architecture is compatible with different fast maximum inner product search algorithms, allows training and inference on resource constrained devices (low memory, low wall-clock time allowance).

Table 4: Comparison of the wall-clock time (in minutes) for ITT-SRP and IOT. We require $N = 2^c$ actions samples at each timestep. Each experiment was run for 100 epochs, and each epoch has 1,000 timesteps. We also provide p-values from paired t-tests, showing ITT-SRP wall-clock time savings are statistically significant.

Environment	Choice of c	# samples / timestep	ITT-SRP	IOT	p-value
HalfCheetah-v1	10	1024	15.15	30.71	2.62e-19
HalfCheetah-v1	11	2028	17.68	57.50	1.81e-20
HalfCheetah-v1	12	4096	20.02	104.51	8.37e-20
HalfCheetah-v1	13	8192	24.17	218.33	4.19e-21
HalfCheetah-v1	14	16384	31.67	421.67	8.72e-22
Walker2d-v1	10	1024	19.71	39.73	7.24e-15
Walker2d-v1	11	2048	20.42	62.52	6.18e-16
Walker2d-v1	12	4096	22.02	106.67	4.72e-17
Walker2d-v1	13	8192	24.77	203.34	4.34e-21
Walker2d-v1	14	16384	29.82	386.67	1.99e-22

ITT-lazy: we update the action tower every 5 iterations. We show in Appendix C that ITT-lazy variant achieves performance comparable to that of regular ITT while achieves further wall-clock time savings, making ITT particularly well-suited for low-resource settings.

5 Conclusion

We presented in this paper a new model for the architectures of the implicit policies, called the Implicit Two-Tower (ITT) model. ITTs provide substantial computational benefits over their regular Implicit One-Tower (IOT) counterparts, yet at the same time they lead to more accurate models (also as compared to the explicit policies). They are also compatible with various hashing techniques providing additional computational gains, especially if very large sets of sampled actions are needed.

References

Peter L. Bartlett. An introduction to reinforcement learning theory: Value function methods. In Shahar Mendelson and Alexander J. Smola (eds.), *Advanced Lectures on Machine Learning,*

Machine Learning Summer School 2002, Canberra, Australia, February 11-22, 2002, Revised Lectures, volume 2600 of *Lecture Notes in Computer Science*, pp. 184–202. Springer, 2002. doi: 10.1007/3-540-36434-X_5. URL https://doi.org/10.1007/3-540-36434-X_5.

Richard Bellman. Some applications of the theory of dynamic programming - A review. *Oper. Res.*, 2(3):275–288, 1954. doi: 10.1287/opre.2.3.275. URL https://doi.org/10.1287/opre.2.3.275.

Bibit Bianchini, Mathew Halm, Nikolai Matni, and Michael Posa. Generalization bounds for implicit learning of nearly discontinuous functions. *arXiv preprint arXiv:2112.06881*, 2021.

Krzysztof Choromanski, Mark Rowland, Vikas Sindhwani, Richard E. Turner, and Adrian Weller. Structured evolution with compact architectures for scalable policy optimization. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 969–977. PMLR, 2018. URL http://proceedings.mlr.press/v80/choromanski18a.html.

Krzysztof Choromanski, Aldo Pacchiano, Jack Parker-Holder, Yunhao Tang, and Vikas Sindhwani. From complexity to simplicity: Adaptive es-active subspaces for blackbox optimization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 10299–10309, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/88bade49e98db8790df275fcebb37a13-Abstract.html.

Krzysztof Choromanski, Haoxian Chen, Han Lin, Yuanzhe Ma, Arijit Sehanobish, Deepali Jain, Michael S. Ryoo, Jake Varley, Andy Zeng, Valerii Likhosherstov, Dmitry Kalashnikov, Vikas Sindhwani, and Adrian Weller. Hybrid random features. *ICLR 2022*, abs/2110.04367, 2021a. URL https://arxiv.org/abs/2110.04367.

Krzysztof Marcin Choromanski, Mark Rowland, and Adrian Weller. The unreasonable effectiveness of structured random orthogonal embeddings. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 219–228, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/bf8229696f7a3bb4700cfddef19fa23f-Abstract.html.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021b. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Krzysztof Marcin Choromanski, Arijit Sehanobish, Han Lin, Yunfan Zhao, Eli Berger, Tetiana Parshakova, Alvin Pan, David Watkins, Tianyi Zhang, Valerii Likhosherstov, et al. Efficient graph field integrators meet point clouds. In *International Conference on Machine Learning*, pp. 5978–6004. PMLR, 2023.

Jason Xiaotian Dou, Alvin Qingkai Pan, Runxue Bao, Haiyi Harry Mao, and Lei Luo. Sampling through the lens of sequential decision making. *arXiv preprint arXiv:2208.08056*, 2022.

Yilun Du, Toru Lin, and Igor Mordatch. Model-based planning with energy-based models. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pp. 374–383. PMLR, 2019. URL http://proceedings.mlr.press/v100/du20a.html.

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

Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pp. 158–168. PMLR, 2021. URL https://proceedings.mlr.press/v164/florence22a.html.

Chris Gaskett, David Wettergreen, and Alexander Zelinsky. Q-learning in continuous state and action spaces. In Norman Y. Foo (ed.), *Advanced Topics in Artificial Intelligence, 12th Australian Joint Conference on Artificial Intelligence, AI '99, Sydney, Australia, December 6-10, 1999, Proceedings*, volume 1747 of *Lecture Notes in Computer Science*, pp. 417–428. Springer, 1999. doi: 10.1007/3-540-46695-9_35. URL https://doi.org/10.1007/3-540-46695-9_35.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 2455–2467, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 2017. URL http://proceedings.mlr.press/v70/haarnoja17a.html.

Sihong He, Shuo Han, and Fei Miao. Robust electric vehicle balancing of autonomous mobility-on-demand system: A multi-agent reinforcement learning approach. *arXiv preprint arXiv:2307.16228*, 2023a.

Sihong He, Songyang Han, Sanbao Su, Shuo Han, Shaofeng Zou, and Fei Miao. Robust multi-agent reinforcement learning with state uncertainty. *Transactions on Machine Learning Research*, 2023b. ISSN 2835-8856. URL https://openreview.net/forum?id=CqTkapZ6H9.

Sihong He, Yue Wang, Shuo Han, Shaofeng Zou, and Fei Miao. A robust and constrained multi-agent reinforcement learning framework for electric vehicle amod systems. *arXiv preprint arXiv:2209.08230*, 2023c.

Bin Huang and Jianhui Wang. Deep-reinforcement-learning-based capacity scheduling for pv-battery storage system. *IEEE Transactions on Smart Grid*, 12(3):2272–2283, 2020.

Bin Huang and Jianhui Wang. Applications of physics-informed neural networks in power systems-a review. *IEEE Transactions on Power Systems*, 38(1):572–588, 2022.

Bin Huang and Jianhui Wang. Adaptive static equivalences for active distribution networks with massive renewable energy integration: A distributed deep reinforcement learning approach. *IEEE Transactions on Network Science and Engineering*, 2023.

Bin Huang, Tianqiao Zhao, Meng Yue, and Jianhui Wang. Bi-level adaptive storage expansion strategy for microgrids using deep reinforcement learning. *IEEE Transactions on Smart Grid*, 2023.

Deepali Jain, Ken Caluwaerts, and Atil Iscen. From pixels to legs: Hierarchical learning of quadruped locomotion. In Jens Kober, Fabio Ramos, and Claire J. Tomlin (eds.), *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pp. 91–102. PMLR, 2020. URL https://proceedings.mlr.press/v155/jain21a.html.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *CoRR*, abs/1806.10293, 2018. URL http://arxiv.org/abs/1806.10293.

Yann LeCun, Sumit Chopra, Raia Hadsell, Fu Jie Huang, and et al. A tutorial on energy-based learning. In *PREDICTING STRUCTURED DATA*. MIT Press, 2006.

Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 1805–1814, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/7634ea65a4e6d9041cfd3f7de18e334a-Abstract.html.

Yurii Nesterov and Vladimir G. Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17:527–566, 2017.

Behnam Neyshabur and Nathan Srebro. On symmetric and asymmetric lshs for inner product search. In *International Conference on Machine Learning*, pp. 1926–1934. PMLR, 2015.

Ninh Pham. Simple yet efficient algorithms for maximum inner product search via extreme order statistics. In Feida Zhu, Beng Chin Ooi, and Chunyan Miao (eds.), *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pp. 1339–1347. ACM, 2021. doi: 10.1145/3447548.3467345. URL https://doi.org/10.1145/3447548.3467345.

Ninh D. Pham. Sublinear maximum inner product search using concomitants of extreme order statistics. *ArXiv*, abs/2012.11098, 2020.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David P. Kreil, Michael K. Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=tL89RnzIiCd.

Ankit Singh Rawat, Jiecao Chen, Felix X. Yu, Ananda Theertha Suresh, and Sanjiv Kumar. Sampled softmax with random fourier features. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 13834–13844, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/e43739bba7cdb577e9e3e4e42447f5a5-Abstract.html.

Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR*, abs/1703.03864, 2017. URL http://arxiv.org/abs/1703.03864.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Anshumali Shrivastava and Ping Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). *Advances in neural information processing systems*, 27, 2014.

Anshumali Shrivastava and Ping Li. Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (MIPS). In Marina Meila and Tom Heskes (eds.), *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI 2015, July 12-16, 2015, Amsterdam, The Netherlands*, pp. 812–821. AUAI Press, 2015a. URL http://auai.org/uai2015/proceedings/papers/96.pdf.

Anshumali Shrivastava and Ping Li. Asymmetric minwise hashing for indexing binary inner products and set containment. In Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi (eds.), *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pp. 981–991. ACM, 2015b. doi: 10.1145/2736277.2741285. URL https://doi.org/10.1145/2736277.2741285.

Siddharth Singi, Zhanpeng He, Alvin Pan, Sandip Patel, Gunnar A Sigurdsson, Robinson Piramuthu, Shuran Song, and Matei Ciocarlie. Decision making for human-in-the-loop robotic agents via uncertainty-aware reinforcement learning. *arXiv preprint arXiv:2303.06710*, 2023.

Yang Song and Diederik P. Kingma. How to train your energy-based models. *CoRR*, abs/2101.03288, 2021. URL https://arxiv.org/abs/2101.03288.

Zhao Song, Ronald Parr, and Lawrence Carin. Revisiting the softmax bellman operator: New benefits and new perspective. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5916–5925. PMLR, 2019. URL http://proceedings.mlr.press/v97/song19c.html.

Narayanan Sundaram, Aizana Turmukhametova, Nadathur Satish, Todd Mostak, Piotr Indyk, Samuel Madden, and Pradeep Dubey. Streaming similarity search over one billion tweets using parallel locality-sensitive hashing. *Proceedings of the VLDB Endowment*, 6(14):1930–1941, 2013.

Richard S. Sutton. Reinforcement learning: Past, present and future. In Bob McKay, Xin Yao, Charles S. Newton, Jong-Hwan Kim, and Takeshi Furuhashi (eds.), *Simulated Evolution and Learning, Second Asia-Pacific Conference on Simulated Evolution and Learning, SEAL'98, Canberra, Australia, November 24-27 1998, Selected Papers*, volume 1585 of *Lecture Notes in Computer Science*, pp. 195–197. Springer, 1998. doi: 10.1007/3-540-48873-1_26. URL https://doi.org/10.1007/3-540-48873-1_26.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998. ISBN 978-0-262-19398-6. URL https://www.worldcat.org/oclc/37293240.

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

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In Dale Schuurmans and Michael P. Wellman (eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pp. 2094–2100. AAAI Press, 2016. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12389.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Christopher J. C. H. Watkins and Peter Dayan. Technical note q-learning. *Mach. Learn.*, 8:279–292, 1992. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International Conference on Machine Learning*, pp. 2635–2644. PMLR, 2016.

Yifei Xu, Jianwen Xie, Tianyang Zhao, Chris Baker, Yibiao Zhao, and Ying Nian Wu. Energy-based continuous inverse optimal control. *IEEE transactions on neural networks and learning systems*, 2022.

Wenhao Yu, Deepali Jain, Alejandro Escontrela, Atil Iscen, Peng Xu, Erwin Coumans, Sehoon Ha, Jie Tan, and Tingnan Zhang. Visual-locomotion: Learning to walk on complex terrains with vision. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1291–1302. PMLR, 2021. URL https://proceedings.mlr.press/v164/yu22a.html.

Hanhan Zhou, Tian Lan, and Vaneet Aggarwal. Pac: Assisted value factorization with counterfactual predictions in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:15757–15769, 2022.

Hanhan Zhou, Tian Lan, and Vaneet Aggarwal. Value functions factorization with latent state information sharing in decentralized multi-agent policy gradients. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2023.

A BACKGROUND

Q-learning: Q-learning methods (Gaskett et al. (1999); Watkins & Dayan (1992); van Hasselt et al. (2016); Kalashnikov et al. (2018); He et al. (2023b); Huang et al. (2023)), that are prominent examples of the off-policy RL algorithms, can be thought of as instantiations of the implicit policies techniques. The Q-function can be interpreted as the negated energy and it has a very special semantics: $Q(\mathbf{s}, \mathbf{a})$ stands for the total reward obtained by an agent applying action $\mathbf{a}$ in state $\mathbf{s}$ and then following optimal policy. Consequently, the training of the (neural network) approximation $\widehat{Q}$ of Q leverages the fact that Q is a fixed point of the so-called *Bellman operator* (Bellman (1954); Song et al. (2019)). Furthermore, learning the Q-function is an off-policy process and the argmin-defined policy is applied only after Q-learning is completed. The setting considered in this paper is more general - the energy E lacks the semantics of the negated Q-function which enables us to bypass the separate off-policy training of E. The algorithms presented in this paper are in fact on-policy.

Energy-based Models: Implicit policies can be viewed as special instantiations of energy-based models (EBMs) (see: LeCun et al. (2006); Song & Kingma (2021) for a comprehensive introduction to EBMs). Several impactful ML architectures have been recently reinterpreted as EBMs. Those include Transformers Vaswani et al. (2017) with their attention modules resembling modern associative memory units (the latter being flagship examples of EBMs Ramsauer et al. (2021) implementing differentiable dictionaries via exponential energies). We mention Transformers here on purpose. In ITTs the energy is the dot product of latent action and negated latent state. Thus, ITTs can be interpreted as learning the cross-attention between the state and action-spaces with actions corresponding to keys and states to queries.

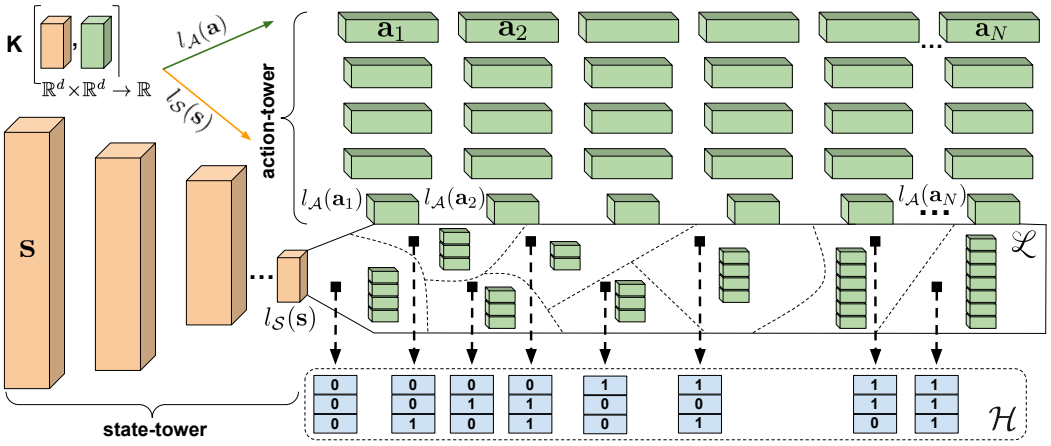

Figure 1: The conceptual description of the Implicit Two-Tower (ITT) stack. The (orange) state-tower and (green) action-tower map states $\mathbf{s}$ and actions $\mathbf{a}$ into their corresponding latent representations $l_{\mathcal{S}}(\mathbf{s}), l_{\mathcal{A}}(\mathbf{a}) \in \mathbb{R}^d$. The latent space $\mathscr{L}$ can be itself partitioned into subregions, for instance via hashing mechanisms, for the sublinear approximate state-action match. This potential partitioning would need to be periodically updated in the training process, but in principle could be frozen in inference (if a fixed set of sampled actions is being applied). The energy-function is defined via a simple kernel $\mathrm{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ acting on the latent representations. Symbol $\mathcal{H}$ refers to the hash space.

B FAST MAXIMUM INNER PRODUCT & BEYOND

B.1 SIGNED RANDOM PROJECTIONS

The formulation from Equation 5 is amenable to the hashing-based relaxation. In this setting set $\mathcal{A}^*$ is partitioned into nonempty subsets: $\mathcal{A}^*_1, ..., \mathcal{A}^*_p$ based on the hash-map: $h : \mathbb{R}^d \rightarrow \mathbb{Z}^m$, where $\mathbb{Z}$ is the set of all integers (a given subset of the partitioning contains actions from $\mathcal{A}^*$ with the same vector-value of the hash-map). Hashing techniques (e.g. locality-sensitive hashing) are applied on the regular basis to speed up nearest neighbor search (NNS). The MIP-formulation at first glance does not look like the NNS, but can be easily transformed to the NNS-formulation (see: Pham (2020)), leading in our case to the new definitions of the latent embeddings corresponding to actions and states:

$$\tilde{l}^{\theta_2}_{\mathcal{A}}(\mathbf{a}) = \left[(l^{\theta_2}_{\mathcal{A}}(\mathbf{a}))^\top, \sqrt{C^2 - \|l^{\theta_2}_{\mathcal{A}}(\mathbf{a})\|^2_2}\right]^\top \qquad \tilde{l}^{\theta_1}_{\mathcal{S}}(\mathbf{s}) = [(l^{\theta_1}_{\mathcal{S}}(\mathbf{s}))^\top, 0]^\top, \tag{8}$$

where C stands for the upper bound on the length of the original latent action-representation (e.g. $C = \max_{\mathbf{a}\in\mathcal{A}^*} \|l^{\theta_2}_{\mathcal{A}}(\mathbf{a})\|_2$). If the nonlinearity $g : \mathbb{R} \rightarrow \mathbb{R}$ used in the last layer of the action-tower satisfies: $|g(x)| \leq B$ for some finite $B > 0$, then one can take: $C = B\sqrt{d}$.

Note that the re-formulation from Equation 8 preserves dot-products, i.e. we trivially have:

$$(\tilde{l}^{\theta_2}_{\mathcal{A}}(\mathbf{a}))^\top \tilde{l}^{\theta_1}_{\mathcal{S}}(\mathbf{s}) = (l^{\theta_2}_{\mathcal{A}}(\mathbf{a}))^\top l^{\theta_1}_{\mathcal{S}}(\mathbf{s}), \tag{9}$$

but it has a critical advantage over the previous one - the latent representations of actions have now exactly the same length $L = C$. Thus the original MIP problem becomes the NNS with the angular distance. To approximate the angular distance, we will apply the Signed Random Projection (SRP) method. The method relies on the linearization of the angular kernel via random feature (RF) map mechanism Choromanski et al. (2017). The angular kernel $\mathrm{K}_{\mathrm{ang}} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is defined as:

$$\mathrm{K}_{\mathrm{ang}}(\mathbf{x}, \mathbf{y}) = 1 - \frac{2\theta_{\mathbf{x},\mathbf{y}}}{\pi}, \tag{10}$$

where $\theta_{\mathbf{x},\mathbf{y}}$ is an angle between $\mathbf{x}$ and $\mathbf{y}$. The key observation is that $\mathrm{K}_{\mathrm{ang}}$ can be rewritten as:

$$\mathrm{K}_{\mathrm{ang}}(\mathbf{x}, \mathbf{y}) = \mathbb{E}\left[\phi(\mathbf{x})^\top \phi(\mathbf{y})\right] \qquad \text{for } \phi(\mathbf{z}) \stackrel{\mathrm{def}}{=} \frac{1}{\sqrt{m}}(\mathrm{sgn}(\omega_1^\top \mathbf{z}), ..., \mathrm{sgn}(\omega_m^\top \mathbf{z}))^\top, \tag{11}$$

where $\omega_1, ..., \omega_m \stackrel{\mathrm{iid}}{\sim} \mathcal{N}(0, \mathbf{I}_d)$. Thus each latent state/action representation can be mapped into the hashed space $\{-1, +1\}^m \subseteq \mathbb{Z}^m$ via the mapping: $\mathbf{z} \stackrel{h}{\rightarrow} (\mathrm{sgn}(\omega_1^\top \mathbf{z}), ..., \mathrm{sgn}(\omega_m^\top \mathbf{z}))^\top$ and in that new space the search can occur based on the Hamming-distance from the hash-buckets corresponding to actions. The computational gains are coming from the fact that during that search, for a given input state $\mathbf{s}$, lots of these buckets (and thus also corresponding sets of sampled actions) will not need to be exercised at all. In our implementation, we construct $\omega_1, ..., \omega_m$ such that their marginal distributions are still Gaussian (thus unbiasedness of the angular kernel estimation is maintained), yet they form a block-orthogonal ensemble. This provides additional variance reduction for any number m of RFs (see: Choromanski et al. (2017)). The ITT-pipeline applying SRPs is schematically presented in Fig. 2

B.2 RANDOM FEATURE TREES

Let us assume that kernel K can be linearized as follows: $\mathrm{K}(\mathbf{x}, \mathbf{y}) = \mathbb{E}[\phi(\mathbf{x})^\top \phi(\mathbf{y})]$ for some (potentially randomized) mapping ϕ. Denote by ψ the positive random feature map mechanism (FAVOR+) from Choromanski et al. (2021b) for linearizing the softmax-kernel (i.e.: $\exp(\mathbf{x}^\top \mathbf{y}) = \mathbb{E}[\psi(\mathbf{x})^\top \psi(\mathbf{y})]$). Without loss of generality, we will assume that the size of the sampled actions set $\mathcal{A}^*$ satisfies: $|\mathcal{A}^*| = 2^k$ for some $k \in \mathbb{N}$. We construct a binary tree $\mathcal{T}$ with nodes corresponding to the subsets of $\mathcal{A}^*$. In the root we put the entire set $\mathcal{A}^*$. The set of actions in each non-leaf node is split into two equal-size parts (uniformly at random) and those are assigned to its two children. Leaves of the tree correspond to singleton-sets of actions.

Action assignment for a given state $\mathbf{s}$ is conducted via the binary search in $\mathcal{T}$ starting at its root. Whenever the algorithm reaches the leaf, its corresponding action is assigned to the input state $\mathbf{s}$. Assume that the algorithm reached non-leaf node v of $\mathcal{T}$. Denote its children as: v_l and v_r and the corresponding action-sets as $\mathcal{A}^*_{v_l}$ and $\mathcal{A}^*_{v_r}$ respectively. For the ITT architecture, the following is true:

Lemma B.1.

$$\mathbb{P}[\widehat{\pi_\theta}(\mathbf{s}) \in \mathcal{A}^*_{v_l} | \widehat{\pi_\theta}(\mathbf{s}) \in \mathcal{A}^*_{v_l} \cup \mathcal{A}^*_{v_r}] = \frac{\psi(\phi(l^{\theta_1}_{\mathcal{S}}(\mathbf{s})))^\top \xi(v_l)}{\psi(\phi(l^{\theta_1}_{\mathcal{S}}(\mathbf{s})))^\top (\xi(v_l) + \xi(v_r))},$$

where $\xi(v) \stackrel{\text{def}}{=} \sum_{a \in \mathcal{A}^*_v} \psi(\phi(l^{\theta_2}_{\mathcal{A}}(\mathbf{a})))$.

The proof is relegated to the appendix. We conclude that in order to decide whether to choose v_l or v_r, the algorithm just needs to sample from the binary distribution with: $p_l = \frac{a}{a+b}, p_r = \frac{b}{a+b}$, where $a = \psi(\phi(l^{\theta_1}_{\mathcal{S}}(\mathbf{s})))^\top \xi(v_l)$, $b = \psi(\phi(l^{\theta_1}_{\mathcal{S}}(\mathbf{s})))^\top \xi(v_r)$. Thus if $\xi(v)$ is computed for every node, this sampling can be conducted in time constant in N.

The total time of assigning the action to the input state is $O(\log(N))$ (rather than linear) in the number of sampled actions (but linear in the number of random features). In practice, the actions do not need to be stored explicitly in the tree and in fact even the tree-structure does not need to be stored explicitly. We call the above tree the *Random Feature Tree* (or RFT) (see also: Choromanski et al. (2021a); Rawat et al. (2019)).

Lazy Action-tower Ppdates: Both considered data structures: SRP- and RFT-based hashes need to be updated every time the parameters of the action-tower are updated, but provide desired speedups between consecutive updates (if the sets of chosen sampled actions do not change). Fortunately, in the ITT-model, the frequency of updates of the action-tower can be completely disentangled from the one for the state-tower. In particular, the action-tower can be updated much less frequently or with frequency decaying in time.

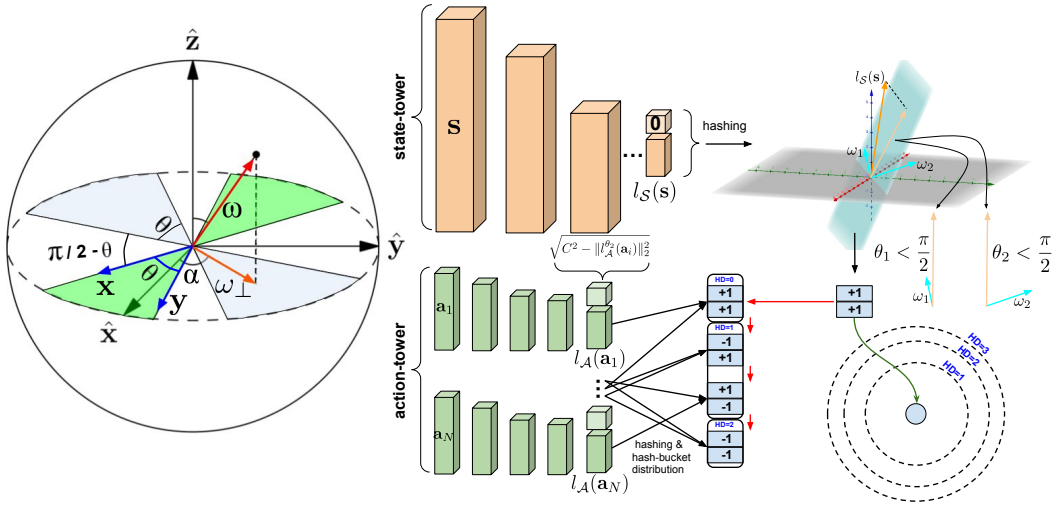

Figure 2: Pictorial description of the Signed Random Projections (SRP) LSH-hashing mechanism that can be applied in the ITT-model, in particular when larger sets of sampled actions are needed. **Left:** Explanation of the SRP-mechanism. SRP relies on the fact that the probability that $\text{sgn}(\omega^\top \mathbf{x})\text{sgn}(\omega^\top \mathbf{y}) < 0$ is the same as the probability that the projection $\omega_\perp$ of $\omega \sim \mathcal{N}(0, \mathbf{I}_d)$ into the subspace spanned by $\{\mathbf{x}, \mathbf{y}\}$ forms angle $< \frac{\pi}{2}$ with one of $\{\mathbf{x}, \mathbf{y}\}$ and $> \frac{\pi}{2}$ with the other one (vector $\omega_\perp$ inside one of the blue regions). That probability is proportional to $\theta_{\mathbf{x},\mathbf{y}}$ since $\omega_\perp$ is also Gaussian **Right:** The latent representation of the state is hashed via the projections onto a random hyperplane spanned by the Gaussian vectors ω_i. The hash's entries are determined based on the angle θ_i formed with vectors ω_i (+1 for $\theta_i < \frac{\pi}{2}$ and -1 for $\theta_i > \frac{\pi}{2}$). Action hash-buckets are sorted by the Hamming distance from the state's hash and the search is conducted in that order (red arrows).

C ADDITIONAL EXPERIMENTAL DETAILS

C.1 NEURAL NETWORK SPECIFICATIONS AND HYPER-PARAMETER TUNING

Hyperparameter Tuning Tables 5 illustrate fine-tuned hyper-parameter σ, which controls the exploration in ES-optimization. For each policy-architecture (ITT, IOT, and explicit) and for each environment, we choose the value of $\sigma \in [0.1, 0.5, 1]$ that provide the highest final average score at the end of the horizon. Similarly, we choose the number of neural network layer in $[1, 2, 3, 4, 5, 6]$ that provide the highest final average score at the end of the horizon. For fair comparison, we ensure the number of trainable parameters of ITTs is upper-bounded by that of IOTs and explicit variants.

Environment	**ITT**	**IOT**	**Explicit**
Swimmer-v2	1	1	1
LunarLanderContinuous-v2	1	1	1
Hopper-v2	1	1	1
HalfCheetah-v2	1	1	0.5
Walker2d-v2	0.5	0.5	0.5
CartPole-v1	1	1	1
MountainCar-v0	1	1	1
Acrobot-v1	1	1	1
MountainCarContinuous-v0	1	1	1
InvertedPendulumBulletEnv-v0	1	1	1
DMCS:FishSwim	0.1	0.1	0.1
DMCS:Swimmer6	0.1	0.1	0.1
DMCS:Swimmer15	0.1	0.1	0.1
DMCS:HopperStand	0.1	0.1	0.1
DMCS:WalkerWalk	0.1	0.1	0.1

Table 5: Fine-tuned hyper-parameter σ, used in ES gradient estimator calculations (Equation 6) for different environments and different policy-architectures.

Environment	**ITT**	**IOT**	**Explicit**
Swimmer-v2	20	22	20
LunarLanderContinuous-v2	20	22	20
Hopper-v2	42	45	42
HalfCheetah-v2	282	288	282
Walker2d-v2	246	252	246
CartPole-v1	6	7	6
MountainCar-v0	4	5	4
Acrobot-v1	8	9	8
MountainCarContinuous-v0	3	4	3
InvertedPendulumBulletEnv-v0	12	13	12
DMCS:FishSwim	2180	2200	2180
DMCS:Swimmer6	2200	2220	2200
DMCS:Swimmer15	3100	3120	3100
DMCS:HopperStand	1980	2000	1980
DMCS:WalkerWalk	2200	2220	2200

Table 6: The dimensionality of the learnable $\theta \in \mathbb{R}^D$ for different environments and different policy-architectures.

Neural Network Specifications Tables 6–7 illustrate the dimension of the learnable $\theta \in \mathbb{R}^D$ and the number of layers in neural networks respectively. The dimensionality of the latent state and action vector as well as the dimensionalities of the hidden layers are set to the dimensionality of the action vector for the $\mathrm{OpenAI\ Gym}$ environments and are equal to 20 for the DMCS environments. We do not use bias terms. We apply Relu activation for all the hidden layers and linear activation on the output layers, with one exception: for the Swimmer-v2 environment, we use linear activation in all

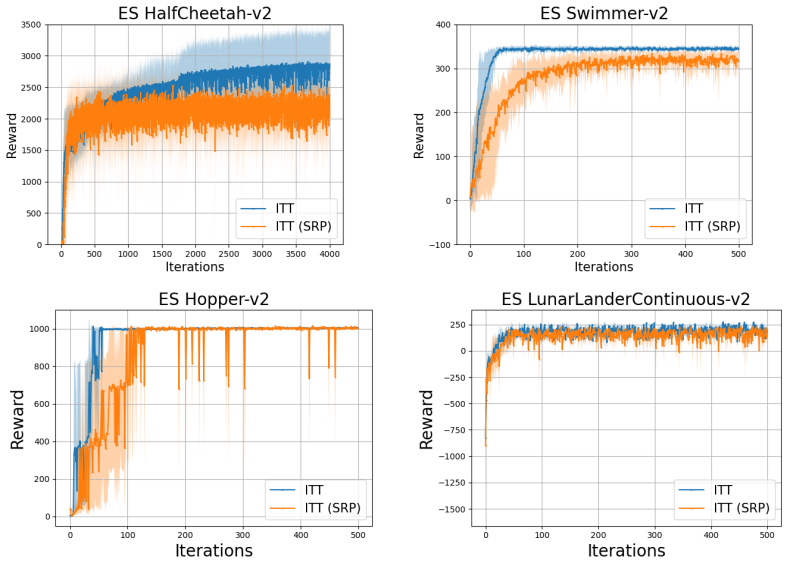

Figure 3: Comparison of vanilla ITT and ITT with Signed Random Projection for fast MIP. We plot average curves obtained from 10 seeds, and we present the 90^{th} percentile and the 10^{th} percentile as shadowed regions. We present results on **additional tasks** in Appendix C.2.

layers and for all the methods (because in this environment only, using linear activation in all layers improves the performance of the baselines).

Environment	**ITT state tower**	**ITT action tower**	**IOT**	**Explicit**
Swimmer-v2	1	1	2	2
LunarLanderContinuous-v2	1	1	2	2
Hopper-v2	1	1	2	2
HalfCheetah-v2	4	2	6	6
Walker2d-v2	3	2	5	5
CartPole-v1	2	1	3	3
MountainCar-v0	2	1	3	3
Acrobot-v1	2	1	3	3
MountainCarContinuous-v0	1	1	2	2
InvertedPendulumBulletEnv-v0	1	1	2	2
DMCS:FishSwim	3	1	5	5
DMCS:Swimmer6	3	1	5	5
DMCS:Swimmer15	3	1	5	5
DMCS:HopperStand	3	1	5	5
DMCS:WalkerWalk	3	1	5	5

Table 7: The number of layers of the neural networks encoding different architectures for different environments.

C.2 ITT-SRP Results

Figure 3 shows the comparison of the regular ITTs with ITTs applying SRPs. The random projection vectors $\omega_1, ..., \omega_m$ have marginal distribution $\mathcal{N}(0, \mathbf{I}_d)$ and are conditioned to be orthogonal. The orthogonality is obtained via the Gram-Schmidt orthogonalization of the iid samples (see: Choromanski et al. (2017)). When m is small, to avoid having too many or too few actions in each action hash-bucket, we calculate $b_i = \text{median}(\{\omega_i^\top \tilde{l}_{\mathcal{A}}^{\theta_1}(\mathbf{a_j})\}_{j=1}^N)$ for each projection vector ω_i, and map the action to the hashed space using $\mathbf{z} \xrightarrow{h} (\text{sgn}(\omega_1^\top \mathbf{z} - b_1), ..., \text{sgn}(\omega_m^\top \mathbf{z} - b_m))^\top$. The number of projection vectors used are $m = 6$ for HalfCheetah-v2, and $m = 3$ for all other environments.

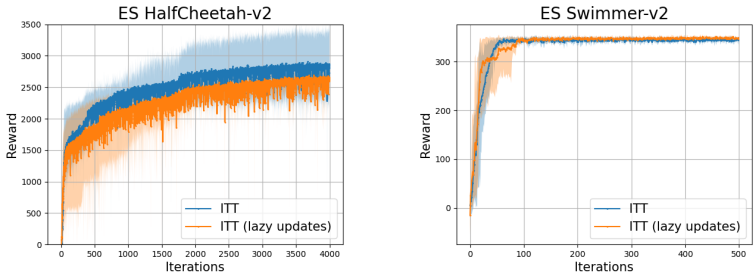

Figure 4: Comparison of vanilla ITT and ITT with lazy action-tower updates, where actions-towers are updated once every 5 iterations. We plot average curves obtained from $s = 10$ seeds, and present the 90^{th} percentile and the 10^{th} percentile using shadowed regions.

C.3 ITT WITH LAZY ACTION UPDATES

Table 8 shows the wall-clock time (in minutes on 30 CPU cores on Google Cloud Compute) for regular ITT and ITT with lazy action updates (**18%** training time reduction for HalfCheetah-v2, **9%** for Walker2d-v2, **24%** for Hopper-v2). Each experiment was run for 100 epochs, and each epoch has 1,000 timesteps. We require 1,000 action samples in each timestep. We conducted paired t-tests on wall-clock times between regular ITT and ITT (lazy). The null hypothesis is that there is no difference between the mean of the wall-clock time. The results show that p-values (for all tasks) are below 0.05, and thus we reject the null hypothesis. We conclude that the wall-clock time savings of ITT (lazy) is statistically significant at a 95% confidence level.

Environment	**regular ITT**	**ITT (lazy)**	**Paired t-test p-value**
HalfCheetah-v2	25.22	20.67	3.31e-8
Walker2d-v2	27.78	25.16	3.19e-2
Hopper-v2	3.19	2.44	2.61e-3

Table 8: Comparison of the wall-clock time (in minutes) for regular ITT and ITT with lazy action updates. Each experiment was run for 100 epochs, and each epoch has 1,000 timesteps. **We also provide p-values from paired t-tests, showing ITT (lazy) wall-clock time savings are statistically significant at a 95% confidence level (all p-values presented are below 0.05).**

D ABLATION STUDIES

D.1 PERFORMANCE AGAINST THE NUMBER OF NEURONS

We present ablation study results on ITT, using different network architectures. The results suggest that ITT consistently outperforms baselines, even when we significantly reduce the number of neurons in the network. In Table 9, we present the number of timesteps to reach given reward thresholds set by previous works(Salimans et al., 2017; Schulman et al., 2015). In Table 10, we provide the number of neurons in the network. In the tables, ITT-1 and ITT-2 stand for variants of ITT with the number of neurons significantly reduced. Notice that ITT-1 has less neurons than ITT, and ITT-2 has less neurons than ITT-1. We observe that the performance of ITT degrades slightly when the number of neurons in the network is decreased.

Environment	**Reward threshold**	**ITT**	**ITT-1**	**ITT-2**	**ES**	**TRPO**
Swimmer-v1	128.25	1.07e+06	1.14e+06	1.30e+06	1.39e+06	4.59e+06
Hopper-v1	877.45	0.69e+05	1.34e+05	1.66e+05	3.83e+05	7.29e+05
Walker2d-v1	957.68	3.16e+05	3.30e+05	3.35e+05	6.43e+05	1.55e+06

Table 9: We present the **number of timesteps** needed to reach given reward thresholds set by previous works (Salimans et al., 2017; Schulman et al., 2015). The results were averaged over 6 random seeds. "ES" stands for the results of explicit policies provided by (Salimans et al., 2017). "ITT-1" and "ITT-2" stand for variants of ITT with the number of neurons significantly reduced.

Environment	ITT	ITT-1	ITT-2
Swimmer-v1	20	18	14
Hopper-v1	42	32	28
Walker2d-v1	246	210	175

Table 10: The dimensionality of the learnable $\theta \in \mathbb{R}^D$ for different environments.

D.2 USE REGULAR ITT IN TRAINING AND ITT-SRP IN INFERENCE

ITT-SRPs can be applied in both training and/or inference. For higher accuracy, one can train with the regular ITT and run inference with ITT-SRP. In Table 11, we present the results for using regular ITT in training and using ITT-SRP in inference. The results show that doing so achieves slightly higher reward than using ITT-SRP in both training and inference.

Training **Inference**	**ITT** **ITT**	**ITT** **ITT-SRP**	**ITT-SRP** **ITT-SRP**	**# iter**
Swimmer-v2	**344.13** $\pm$ 5.18	<u>341.38</u> $\pm$ 7.93	333.84 $\pm$ 7.26	500
LunarLanderC-v2	**157.38** $\pm$ 71.81	<u>154.41</u> $\pm$ 71.01	149.15 $\pm$ 69.27	500
Hopper-v2	**1007.95** $\pm$ 3.66	999.12 $\pm$ 11.39	<u>1000.54</u> $\pm$ 7.74	500
HalfCheetah-v2	**2866.02** $\pm$ 416.81	<u>2690.64</u> $\pm$ 139.05	2565.74 $\pm$ 143.15	4000

Table 11: We present final average scores over $s = 5$ random seeds together with their std. The best architecture is in bold font and the second best is underscored. The "LunarLanderC-v2' stands for the LunarLanderContinuous-v2 environment.

E PAIRED T-TEST RESULTS

To demonstrate that ITT achieves higher scores than IOT and explicit policies, we provide paired t-test results as evidence of statistical significance.

In Table 12, we present p-values from paired t-tests of final scores of different methods.

ITT vs IOT: ITT achieves significantly higher scores than IOT on all tasks except Walker2d-v2, where ITT is the second best among the three architectures. The p-values (ITT & IOT paired t-test) are below 0.05 for all tasks except Half-Cheetah-v2. Thus, the null hypothesis (no difference between the means of IOT and ITT final score) is rejected given significance level 0.05. We conclude that there is statistically significant difference between the means of final returns of ITT and IOT.

ITT vs Explicit: ITT achieves significantly higher scores than explicit policies on more difficult tasks (Hopper-v2, HalfCheetah-v2, Walker2d-v2). The p-values (ITT & IOT paired t-test) are below 0.05 for Hopper-v2 and HalfCheetah-v2. Thus, the null hypothesis (no difference between the means of IOT and ITT final score) is rejected given significance level 0.05. We conclude that there is statistically significant difference between the means of final returns of ITT and explicit policies on Hopper-v2 and HalfCheetah-v2, which are the more difficult tasks.

For simpler tasks, we also look at the number of iterations needed to achieve given reward thresholds. The reward thresholds are set at the 90% of final average return achieved by IOT.

In Table 13, we present p-values from paired t-tests of the number of iterations needed.

ITT vs IOT: for most environments, the p-values are below 0.05. Thus, the null hypothesis (no difference between the means of IOT and ITT number of iterations to reach given reward threshold) is rejected given significance level 0.05. We conclude that there is statistically significant difference between the number of iterations needed to reach given reward thresholds.

ITT vs Explicit: we do not include explicit in this comparison, because their performance are close on simpler tasks. For more difficult tasks (Hopper-v2, HalfCheetah-v2, Walker2d-v2), Explicit cannot even reach 80% of final average return of ITT.

Environment	ITT & IOT p-value	ITT & Explicit p-value	ITT return	IOT return	Explicit Return
Swimmer-v2	7.09e-6	2.65e-1	344.13	75.54	**347.67**
LunarLanderContinuous-v2	9.73e-6	1.90e-1	**157.38**	-72.72	62.85
Hopper-v2	8.05e-9	5.68e-9	**2670.37**	1036.52	1060.89
HalfCheetah-v2	1.69e-1	2.48e-4	**2866.02**	2696.29	1845.64
Walker2d-v2	2.89e-3	1.63e-1	1897.87	**2909.85**	1346.93
MountainCar-v0	4.28e-05	5.27e-1	**-113.50**	-200	-143.40
MountainCarContinuous-v0	4.08e-3	4.38e-2	**89.17**	25.96	52.34
InvertedPendulumBulletEnv-v0	2.86e-18	N/A	**1000.00**	27.10	**1000.00**

Table 12: We present p-values from paired t-tests of returns of different methods. We underline results that are statistically significant, at a 95% confidence level. The p-values are presented in three significant figures. We also provide final average returns. The best architecture is in bold font.

Environment	ITT & IOT p-value	ITT # iter	IOT # iter	Reward Threshold
Swimmer-v2	2.96e-05	10.80	420.10	67.98
LunarLanderContinuous-v2	2.02e-07	12.20	386.80	-65.44
MountainCar-v0	1.52e-07	733.80	2000.00	-180.0
MountainCarContinuous-v0	6.38e-2	75.80	125.00	23.36
InvertedPendulumBulletEnv-v0	1.12e-07	9.50	112.10	24.39

Table 13: We present p-values from paired t-tests of the number of iterations used by each method to achieve given reward threshold. We underline results that are statistically significant, at a 95% confidence level. The p-values are presented in three significant figures. The reward thresholds are set at the 90% of final average return achieved by IOT.

F Theoretical Results

F.1 Proof of results in in Section 2

Proof of Lemma B.1.

$$
\begin{aligned}
&\mathbb{P}[\widehat{\pi_\theta}(\mathbf{s}) \in \mathcal{A}^*_{v_l} | \widehat{\pi_\theta}(\mathbf{s}) \in \mathcal{A}^*_{v_l} \cup \mathcal{A}^*_{v_r}] \\
=&\frac{\sum_{\mathbf{a}\in\mathcal{A}^*_{v_l}} \exp(-E_\theta(\mathbf{s},\mathbf{a}))}{\sum_{\mathbf{a}\in\mathcal{A}^*_{v_l}\cup\mathcal{A}^*_{v_r}} \exp(-E(\mathbf{s},\mathbf{a}))} \\
=&\frac{\sum_{\mathbf{a}\in\mathcal{A}^*_{v_l}} \exp\left\{\mathrm{K}\left(l^{\theta_1}_{\mathcal{S}}(\mathbf{s}), l^{\theta_2}_{\mathcal{A}}(\mathbf{a})\right)\right\}}{\sum_{\mathbf{a}\in\mathcal{A}^*_{v_l}\cup\mathcal{A}^*_{v_r}} \exp\left\{\mathrm{K}\left(l^{\theta_1}_{\mathcal{S}}(\mathbf{s}), l^{\theta_2}_{\mathcal{A}}(\mathbf{a})\right)\right\}} \\
\approx&\frac{\sum_{\mathbf{a}\in\mathcal{A}^*_{v_l}} \exp\left(\phi(l^{\theta_1}_{\mathcal{S}}(\mathbf{s}))^\top \phi(l^{\theta_2}_{\mathcal{A}}(\mathbf{a}))\right)}{\sum_{\mathbf{a}\in\mathcal{A}^*_{v_l}\cup\mathcal{A}^*_{v_r}} \exp\left(\phi(l^{\theta_1}_{\mathcal{S}}(\mathbf{s}))^\top \phi(l^{\theta_2}_{\mathcal{A}}(\mathbf{a}))\right)} \\
\approx&\frac{\psi(\phi(l^{\theta_1}_{\mathcal{S}}(\mathbf{s})))^\top \sum_{\mathbf{a}\in\mathcal{A}^*_{v_l}} \psi(\phi(l^{\theta_2}_{\mathcal{A}}(\mathbf{a})))}{\psi(\phi(l^{\theta_1}_{\mathcal{S}}(\mathbf{s})))^\top \sum_{\mathbf{a}\in\mathcal{A}^*_{v_l}\cup\mathcal{A}^*_{v_r}} \psi(\phi(l^{\theta_2}_{\mathcal{A}}(\mathbf{a})))} \\
=&\frac{\psi(\phi(l^{\theta_1}_{\mathcal{S}}(\mathbf{s})))^\top \xi(v_l)}{\psi(\phi(l^{\theta_1}_{\mathcal{S}}(\mathbf{s})))^\top (\xi(v_l)+\xi(v_r))},
\end{aligned}
\tag{12}
$$

□

F.2 Proof of results in Section 3

We start with introducing notations that help us simplify the proofs.

Definition F.1 (AT and FD ES-gradient estimator)**.** The antithetic ES-gradient estimator and the forward finite difference ES-gradient estimator applying orthogonal samples are defined as

$$\widehat{\nabla}_M^{\mathrm{AT,\,ort}} F_\sigma(\theta) \coloneqq \frac{1}{2\sigma M} \sum_{i=1}^{M} F^{AT(i)} \quad \text{where} \quad F^{AT(i)} \coloneqq F\left(\theta + \sigma\varepsilon_i\right)\varepsilon_i - F\left(\theta - \sigma\varepsilon_i\right)\varepsilon_i \tag{13}$$

$$\widehat{\nabla}_M^{\mathrm{FD,\,ort}} F_\sigma(\theta) \coloneqq \frac{1}{\sigma M} \sum_{i=1}^{M} F^{FD(i)} \quad \text{where} \quad F^{FD(i)} \coloneqq F\left(\theta + \sigma\varepsilon_i\right)\varepsilon_i - F\left(\theta\right)\varepsilon_i, \tag{14}$$

where $(\varepsilon_i)_{i=1}^M$ have marginal distribution $\mathcal{N}(\mathbf{0}, I_D)$, and $(\varepsilon_i)_{i=1}^M$ are conditioned to be pairwise-orthogonal.

Definition F.2 (Gaussian smoothing)**.** The Gaussian smoothing of $F(x)$ is defined as

$$F_\sigma(\theta) = \frac{1}{\kappa} \int F(\theta + \sigma\varepsilon) e^{-\frac{1}{2}||\varepsilon||_2^2} d\varepsilon \quad \text{, where } \kappa = (2\pi)^{d/2}, \tag{15}$$

and its gradient is

$$\nabla F_\sigma(\theta) = \frac{1}{\sigma\kappa} \int F(\theta + \sigma\varepsilon) e^{-\frac{1}{2}||\varepsilon||_2^2} \varepsilon d\varepsilon. \tag{16}$$

Assumption F.3. Assume $F(\cdot)$ is quadratic. Under this assumption, the gradient $\nabla F(\theta)$ and the Hessian $\nabla^2 F(\theta)$ exist for any $\theta \in \mathbb{R}^D$, and

$$F(\theta + \sigma\epsilon) = F(\theta) + \sigma \nabla F(\theta)^\top \epsilon + \frac{\sigma^2}{2} \varepsilon^\top \nabla^2 F(\theta) \varepsilon.$$

Theorem F.4. *Suppose Assumption F.3 holds. The mean squared error of the AT ES-gradient estimator applying orthogonal samples is*

$$\begin{aligned} \mathrm{MSE}\left(\hat{\nabla}_M^{\mathrm{AT,ort}} F_\sigma(\theta)\right) &\coloneqq \mathbb{E}\left[\left\|\hat{\nabla}_N^{\mathrm{AT,ort}} F_\sigma(\theta) - \nabla F_\sigma(\theta)\right\|_2^2\right] \\ &= \frac{1}{M}\mathbb{E}\left[\left\|(\nabla F(\theta)^\top \varepsilon)\varepsilon\right\|_2^2\right] - \left\|\nabla F_\sigma(\theta)\right\|_2^2. \end{aligned}$$

The mean squared error of the FD ES-gradient estimator applying orthogonal samples is

$$\begin{aligned} \mathrm{MSE}\left(\hat{\nabla}_N^{\mathrm{FD,ort}} F_\sigma(\theta)\right) &\coloneqq \mathbb{E}\left[\left\|\hat{\nabla}_N^{\mathrm{FD,ort}} F_\sigma(\theta) - \nabla F_\sigma(\theta)\right\|_2^2\right] \\ &= \frac{1}{M}\mathbb{E}\left[\left\|(\nabla F(\theta)^\top \varepsilon + \frac{\sigma^2}{2}\varepsilon^\top \nabla^2 F(\theta)\varepsilon)\varepsilon\right\|_2^2\right] - \left\|\nabla F_\sigma(\theta)\right\|_2^2. \end{aligned}$$

Remark F.5. Theorem F.4 can be extended to **general functions** $F(\cdot)$. Classical results on Gaussian smoothing gradient estimators, which motivate the use of ES-gradient estimators, require the objective to be twice continuously differentiable Nesterov & Spokoiny (2017). Their error bound results rely on second order Taylor expansion. Our results for quadratic objectives can be generalized to non-quadratic functions, by imposing twice continuously differentiable assumptions as in Nesterov & Spokoiny (2017) and taking a second order Taylor expansion of a general function, after which we bound the residual terms using the smoothness assumption. One may also take higher order Taylor polynomials of $F(\cdot)$ and bound the residual terms under suitable regularity conditions.

Since we have orthogonal samples, we have the following Lemma.

Lemma F.6. *Assume $M \leq D$, we have*

$$\begin{aligned} \mathrm{MSE}\left(\hat{\nabla}_M^{\mathrm{AT,ort}} F_\sigma(\theta)\right) =& \frac{D+2}{M}||\nabla F(\theta)||_2^2 - \left\|\nabla F_\sigma(\theta)\right\|_2^2 \\ \mathrm{MSE}\left(\hat{\nabla}_M^{\mathrm{FD,ort}} F_\sigma(\theta)\right) =& \frac{D+2}{M}||\nabla F(\theta)||_2^2 + \frac{(D+4)\sigma^4}{4M}||\nabla^2 F(\theta)||_F^2 \\ &+ \frac{(D+2)\sigma^4}{M}\left(\sum_{i=1}^{D} \nabla^2 F(\theta)_{ii}^2\right) - \left\|\nabla F_\sigma(\theta)\right\|_2^2, \end{aligned}$$

and therefor

$$\begin{aligned}&\mathrm{MSE}\left(\hat{\nabla}_M^{\mathrm{FD,ort}} F_\sigma(\theta)\right) - \mathrm{MSE}\left(\hat{\nabla}_M^{\mathrm{AT,ort}} F_\sigma(\theta)\right)\\ =&\frac{(D+4)\sigma^4}{4M}||\nabla^2 F(\theta)||_F^2 + \frac{(D+2)\sigma^4}{M}\left(\sum_{i=1}^{D}\nabla^2 F(\theta)_{ii}^2\right).\end{aligned}$$

The proofs of Theorem F.4 and Lemma F.6 are at the end of Section F.

Remark F.7. Suppose Assumption F.3 holds. We observe that evaluating $\hat{\nabla}_M^{\mathrm{AT,ort}} F_\sigma(\theta)$ requires 2M queries of $F(\cdot)$ and evaluating $\hat{\nabla}_M^{\mathrm{FD,ort}} F_\sigma(\theta)$ requires only M+1 queries of $F(\cdot)$. Consequently, FD-gradient estimator is preferred when $\sigma^4||\nabla^2 F(\theta)||_F^2 \ll ||\nabla F(\theta)||_2^2$, and AT-gradient estimator is preferred when $\sigma^4||\nabla^2 F(\theta)||_F^2 \gg ||\nabla F(\theta)||_2^2$, which is highly likely when σ is large.

proof of Theorem F.4. **AT ES-gradient estimator.**

$$\begin{aligned}&\mathrm{MSE}\left(\hat{\nabla}_N^{\mathrm{AT,ort}} F_\sigma(\theta)\right) = \mathbb{E}\left[\left\|\frac{1}{M}\sum_{i=1}^{M} F^{AT(i)} - \nabla F_\sigma(\theta)\right\|_2^2\right]\\ =&\mathbb{E}\left[\left\|\frac{1}{M}\sum_{i=1}^{M} F^{AT(i)}\right\|_2^2\right] - \|\nabla F_\sigma(\theta)\|_2^2\end{aligned}$$

The first term is

$$\begin{aligned}&\mathbb{E}\left[\left\|\frac{1}{M}\sum_{i=1}^{M} F^{AT(i)}\right\|_2^2\right] = \frac{1}{M^2}\left(\sum_{i=1}^{M}\mathbb{E}\left[\left\|F^{AT(i)}\right\|_2^2\right] + \sum_{i\neq j}\mathbb{E}\left[\left\langle F^{AT(i)}, F^{AT(j)}\right\rangle\right]\right)\\ =&\frac{1}{M^2}\left(\sum_{i=1}^{M}\mathbb{E}\left[\left\|F^{AT(i)}\right\|_2^2\right]\right) = \frac{1}{M^2}\left(\sum_{i=1}^{M}\mathbb{E}\left[\left\|\frac{1}{2\sigma}(F(\theta+\sigma\varepsilon_i)\varepsilon_i - F(\theta-\sigma\varepsilon_i)\varepsilon_i)\right\|_2^2\right]\right)\\ =&\frac{1}{M}\mathbb{E}\left[\left\|\frac{1}{2\sigma}(F(\theta+\sigma\epsilon)\epsilon - F(\theta-\sigma\epsilon)\epsilon)\right\|_2^2\right] = \frac{1}{M}\mathbb{E}\left[\left\|(\nabla F(\theta)^\top\varepsilon)\varepsilon\right\|_2^2\right],\end{aligned}$$

where the second equality is by orthogonality of ε_i; the fourth equality is because $\varepsilon_i are$ i.i.d.; the last equality is by Assumption F.3.

FD ES-gradient estimator. For simplicity of presentation, we abbreviate the first few steps, which are the same as that of the AT ES-gradient estimator.

$$\begin{aligned}&\frac{1}{M^2}\left(\sum_{i=1}^{M}\mathbb{E}\left[\left\|F^{FD(i)}\right\|_2^2\right]\right) = \frac{1}{M^2}\left(\sum_{i=1}^{M}\mathbb{E}\left[\left\|\frac{1}{\sigma}(F(\theta+\sigma\varepsilon_i)\varepsilon_i - F(\theta)\varepsilon_i)\right\|_2^2\right]\right)\\ =&\frac{1}{M}\mathbb{E}\left[\left\|\frac{1}{\sigma}(F(\theta+\sigma\epsilon)\epsilon - F(\theta)\epsilon)\right\|_2^2\right] = \frac{1}{M}\mathbb{E}\left[\left\|\varepsilon(\nabla F(\theta)^\top\varepsilon + \frac{\sigma^2}{2}\varepsilon^\top\nabla^2 F(\theta)\varepsilon)\right\|_2^2\right],\end{aligned}$$

where the second equality is because ε_i are i.i.d., and the third equality is by Assumption F.3. □

Proof of Lemma F.6. **AT ES-gradient estimator.**

$$\begin{aligned}&\mathbb{E}\left[\left\|(\nabla F(\theta)^\top\varepsilon)\varepsilon\right\|_2^2\right] = \sum_{i,j,k}\nabla F(\theta)_i\nabla F(\theta)_j\mathbb{E}\left[\varepsilon_i\varepsilon_j\varepsilon_k^2\right] = \sum_{i,k}\nabla F(\theta)_i^2\mathbb{E}\left[\varepsilon_i^2\varepsilon_k^2\right]\\ =&\sum_{i=1}^{D}\nabla F(\theta)_i^2\mathbb{E}\left[\varepsilon_i^4\right] + \sum_{i=1}^{D}\nabla F(\theta)_i^2\sum_{k\neq i}\mathbb{E}\left[\varepsilon_i^2\varepsilon_k^2\right] = (D+2)||\nabla F(\theta)||_2^2,\end{aligned}$$

where the second equality is because odd moments of Gaussian r.v.s are zero. By Theorem F.4, we have

$$\begin{aligned}&\mathrm{MSE}\left(\hat{\nabla}_M^{\mathrm{AT,ort}} F_\sigma(\theta)\right) \coloneqq \mathbb{E}\left[\left\|\hat{\nabla}_N^{\mathrm{AT,ort}} F_\sigma(\theta) - \nabla F_\sigma(\theta)\right\|_2^2\right] \\ =&\frac{1}{M}\mathbb{E}\left[\left\|(\nabla F(\theta)^\top \varepsilon)\varepsilon\right\|_2^2\right] - \|\nabla F_\sigma(\theta)\|_2^2 = \frac{D+2}{M}||\nabla F(\theta)||_2^2 - \|\nabla F_\sigma(\theta)\|_2^2 .\end{aligned}$$

FD ES-gradient estimator.

$$\begin{aligned}&\mathbb{E}\left[\left\|\varepsilon(\nabla F(\theta)^\top \varepsilon + \frac{\sigma^2}{2}\varepsilon^\top \nabla^2 F(\theta)\varepsilon)\right\|_2^2\right] \\ =&\mathbb{E}\left[\left\|\left(\sum_{i=1}^D \nabla F(\theta)_i \varepsilon_i + \frac{\sigma^2}{2}\sum_{i=1}^D\sum_{j=1}^D \varepsilon_i \nabla^2 F(\theta)_{ij}\varepsilon_j\right)\varepsilon\right\|_2^2\right] \\ =&\mathbb{E}\left[\sum_{k=1}^D \varepsilon_k^2\left(\sum_{i=1}^D \nabla F(\theta)_i \varepsilon_i + \frac{\sigma^2}{2}\sum_{i=1}^D\sum_{j=1}^D \nabla^2 F(\theta)_{ij}\varepsilon_j\right)^2\right] \\ =&\sum_{i,j,k} \nabla F(\theta)_i \nabla F(\theta)_j \mathbb{E}\left[\varepsilon_i\varepsilon_j\varepsilon_k^2\right] + \sum_{i,j,k,l}\mathbb{E}\left[\varepsilon_k^2 \nabla F(\theta)_i \varepsilon_i \sigma^2 \varepsilon_j \varepsilon_l \nabla^2 F(\theta)_{jl}\right] \\ &+\sum_{i,j,k}\mathbb{E}\left[\varepsilon_k^2 \frac{\sigma^4}{4}\varepsilon_i^2\varepsilon_j^2 \nabla^2 F(\theta)_{ij}^2\right],\end{aligned}$$

The second term above equals zero, because the odd moments of Gaussian random variables are zero (in a degree 5 polynomial of Gaussian r.v.s, one term must be raised to an odd power); the first term equals $(D+2)||\nabla F(\theta)||_2^2$ by the same argument as for the AT ES-gradient estimator; the third term is

$$\begin{aligned}&\sum_{i,j,k}\mathbb{E}\left[\varepsilon_k^2\frac{\sigma^4}{4}\varepsilon_i^2\varepsilon_j^2\nabla^2 F(\theta)_{ij}^2\right] = \frac{\sigma^4}{4}\left(15\sum_i \nabla^2 F(\theta)_{ii}^2\right) + \frac{\sigma^4}{4}3(D-1)\sum_i \nabla^2 F(\theta)_{ii}^2 \\ &+\frac{\sigma^4}{4}3\sum_i\sum_{j\neq i}\nabla^2 F(\theta)_{ij}^2 + \frac{\sigma^4}{4}3\sum_j\sum_{i\neq j}\nabla^2 F(\theta)_{ij}^2 + \frac{\sigma^4}{4}(D-2)\left(\sum_i\sum_{j\neq i}\nabla^2 F(\theta)_{ij}^2\right) \\ =&\frac{(D+4)\sigma^4}{4}\|\nabla^2 F(\theta)\|_F^2 + \frac{(8+4D)\sigma^4}{4}\sum_{i=1}^D \nabla^2 F(\theta)_{ii}^2,\end{aligned}$$

where in the first equality, the five terms correspond to $i=j=k, i=j\neq k, i=k\neq j, j=k\neq i$ and distinct i,j,k respectively.

□