Implicit Two-Tower Policies

Yunfan Zhao*1  Qingkai Pan*2  Krzysztof Choromanski*3,4
Deepali Jain3  Vikas Sindhwani3

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, we achieve two main goals: substantial computational gains and better performance. 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 are particularly suited for black-box/evolutionary optimization and the corresponding policy training algorithms outperform their vanilla unstructured implicit counterparts as well as commonly used explicit policies. We complement our analysis by showing how techniques such as hashing and lazy tower updates, critically relying on the two-tower structure of ITTs, can be applied to obtain additional computational gains.

1 INTRODUCTION & RELATED WORK

We consider the problem of training a policy \( \pi_\theta : S \rightarrow A \), parameterized by learnable \( \theta \in \mathbb{R}^D \), for a reinforcement learning (RL) agent (Sutton and Barto (1998); Sutton (1998); Bartlett (2002); Singi et al. (2023); Zhou et al. (2022); He et al. (2023a); Huang and Wang (2020)). The policy is a potentially stochastic mapping from the state-space (\( S \)) to the action-space (\( A \)), either continuous or discrete. The objective is to maximize the expected total reward \( R \) defined as a possibly discounted sum of the partial rewards \( r_i(s_i, a_i, s_{i+1}) \) for the transition from \( s_i \in S \) to \( s_{i+1} \in S \) via \( a_i \in A \). The transition function: \( T : S \times A \rightarrow S \) as well as the partial reward function: \( r : S \times A \times S \rightarrow \mathbb{R} \) (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 \( (s_0, a_0, s_1, a_1, ..., s_T) \) of states visited by the agent intertwined with the actions proposed by \( \pi_\theta \), the rollout of an agent.

The most common way of encoding policy mapping \( \pi_\theta \) 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 and Schmidhuber (2018); Choromanski et al. (2018); Yu et al. (2021); Jain et al. (2020); Huang and 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 \( E_\theta : S \times A \rightarrow \mathbb{R} \) 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 \( a^*(s) \) for a given state \( s \) is chosen as a solution to the following energy-minimization problem:

\[
 a^*(s) = \pi_\theta(s) = \arg\min_{a \in A} E_\theta(s, a). \tag{1}
\]

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 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 $L$. The negated energy $-E$ is then defined as a relatively simple kernel $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ acting on the state/action latent representations (e.g. dot-product or softmax kernel). Such a representation has several computational advantages:

- it enables reusing computations for a fixed set of actions; computational gains are present even if actions are sampled at every transition since in ITTs (as opposed to IOTs) action-processing is completely disentangled from the expensive state-transitions of the environment and thus latent representations of actions can be pre-computed,
- it allows for action and state towers to be updated in different rates in training (e.g. less-frequent lazy action-tower updates) (see: Sec. 2.3),
- it is compatible with various approximate sublinear-time algorithms for solving the argmin-problem in its new two-tower form (see: Sec 2.3.1).

ITTs can in particular apply a rich set of techniques for solving the maximum inner product (MIP) problem Pham (2021); Shrivastava and Li (2015b,a); 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). 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 A on 15 environments taken from OpenAI Gym and DeepMind Control Suite. ITTs can be applied for both: discrete and continuous action-spaces.

The implicit policies are intrinsically related to several classes of methods developed for machine learning (ML) and robotics. We review some of them below.

**Q-learning:** Q-learning methods Gaskett et al. (1999); Watkins and Dayan (1992); van Hasselt et al. (2016); Kalashnikov et al. (2018); He et al. (2023); 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: $Q(s,a)$ stands for the total reward obtained by an agent applying action $a$ in state $s$ and then following optimal policy. Consequently, the training of the (neural network) approximation $\hat{Q}$ 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 and Kingma (2021) for a comprehensive introduction to EBMs). Several impactful ML architectures have been recently reinterpreted as EBMs. Those include Transformers
Our Main Contributions Are:

- We propose a new class of structured reinforcement learning architecture, implicit two-tower policies. We demonstrate the new architecture achieves stronger performance than existing implicit policies and their explicit counterparts.
- We adapt fast maximum inner product search algorithms to solve the energy maximization problem in implicit policies. Thus, we resolve the exponential sample complexity problem for implicit policies and allow implicit policies to scale.
- 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 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 $\pi_\theta : S \rightarrow A$ from the state-space $S \subseteq \mathbb{R}^s$ to the action-space $A \subseteq \mathbb{R}^a$, given as follows for the learnable $\theta \in \mathbb{R}^D$:

$$\pi_\theta(s) = \arg\min_{a \in A} E_\theta(s, a)$$

(2)

Here $E_\theta : S \times A \rightarrow \mathbb{R}$ 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: input = [s, a]. Solving optimization problem from Eq. 2 directly is usually prohibitively expensive. Thus instead sampling strategies are often used. For the selected set $A^* = \{a_1, ..., a_N\}$ of sampled actions (usually uniformly at random from $A$), the algorithm approximates $\pi_\theta(s)$ as: $\hat{\pi}_\theta(s) = \arg\min_{a \in A^*} E_\theta(s, a)$ 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 $a \in A^*$ is sampled from the following softmax-distribution defined on $A^*$ (the relaxation allows back-propagating through the action-selection modules):

$$P[\hat{\pi}_\theta(s) = a_i] = \frac{\exp(-E_\theta(s, a_i))}{\sum_{a \in A^*} \exp(-E_\theta(s, a))}$$

(3)
Implicit Two-Tower Policies

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

In the implicit two-tower (ITT) model the energy is defined as:

$$E_\theta(s, a) = -K(l^0_\mathcal{S}(s), l^0_A(a))$$  \hspace{1cm} (4)

for the state-tower mapping: $l^0_\mathcal{S}(s) : \mathbb{R}^s \to \mathcal{L} \subseteq \mathbb{R}^d$ and action-tower mapping: $l^0_A(a) : \mathbb{R}^a \to \mathcal{L} \subseteq \mathbb{R}^d$, 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 $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. 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. $K(x, y) = f(x^\top y)$ for some $f : \mathbb{R} \to \mathbb{R}$. Those include dot-product kernel, where $f$ is a linear function as well as the softmax-kernel, where $f(z) \triangleq \exp(z)$. For those kernels the corresponding argmax-problems are trivially equivalent and reduce to the maximum-inner-product (MIP) search $\text{Shrivastava and Li (2014); Neyshabur and Srebro (2015); Sundaram et al. (2013), but the}$ softmax-distributions differ. For a fixed sampled set of actions $\mathcal{A}^* = \{a_1, ..., a_N\}$, the MIP problem can be solved particularly efficiently as follows:

$$\tilde{a}_{\theta_1, \theta_2}(s) = \arg\max_{a \in \mathcal{A}^*} (l^0_\mathcal{S}(s) \cdot l^0_A(a)),$$  \hspace{1cm} (5)

where the $i^{th}$ rows of the matrix $l^0_\mathcal{S}(\mathcal{A}^*) \in \mathbb{R}^N \times d$ is given as $l^0_\mathcal{S}(a_i)$. 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: $O(N(d + T_S))$, where $T_S$ is the time needed to compute latent representation $l^0_\mathcal{S}(s)$ of $s$.

2.3 Fast Maximum Inner Product & Beyond

2.3.1 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: $h : \mathbb{R}^d \to \mathbb{Z}_m$, 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:

$$l^0_\mathcal{S}(s) = [l^0_\mathcal{S}(s) \cdot l^0_A(a)]^\top,$$  \hspace{1cm} (6)

where $C$ stands for the upper bound on the length of the original latent action-representation (e.g. $C = \max_{a \in \mathcal{A}^*} \|l^0_A(a)\|_2$). If the nonlinearity $g : \mathbb{R} \to \mathbb{R}$ 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 6 preserves dot-products, i.e. we trivially have:

$$(l^0_\mathcal{S}(a))^\top l^0_\mathcal{S}(s) = (l^0_\mathcal{S}(a))^\top l^0_\mathcal{S}(s),$$  \hspace{1cm} (7)

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 $K_{\text{ang}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is defined as:

$$K_{\text{ang}}(x, y) = 1 - \frac{2 \theta_{x,y}}{\pi},$$  \hspace{1cm} (8)

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

$$K_{\text{ang}}(x, y) = \mathbb{E} [\phi(x)^\top \phi(y)]$$

for $\phi(z) \triangleq \frac{1}{m} (\text{sgn}(\omega_1^\top z), ..., \text{sgn}(\omega_m^\top z))^\top$,  \hspace{1cm} (9)

where $\omega_1, ..., \omega_m \overset{iid}{\sim} \mathcal{N}(0, I_d)$. Thus each latent state/action representation can be mapped into the hashed space $\{-1, +1\}^m \subseteq \mathbb{Z}^m$ via the mapping:

$$h \mapsto (\text{sgn}(\omega_1^\top h), ..., \text{sgn}(\omega_m^\top h))^\top$$

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 $s$, 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 unbiased-ness of the angular kernel estimation is maintained), yet
they form a block-orthogonal ensemble. This provides additional variance reduction for any number \( n \) of RFs (see: Choromanski et al. (2017)). The ITT-pipeline applying SRPs is schematically presented in Fig. 2.

### 2.3.2 Random Feature Trees

Let us assume that kernel \( K \) can be linearized as follows:

\[
K(x, y) = E[\phi(x)\psi(y)]
\]

for some (potentially randomized) mapping \( \phi \). Denote by \( \psi \) the positive random feature map mechanism (FAVOR+) from Choromanski et al. (2021b) for linearizing the softmax-kernel (i.e.: \( K(x, y) = Exp(\langle x, y \rangle) \)). Without loss of generality, we will assume that the size of the sampled actions set \( \mathcal{A}^\ast \) satisfies: \( | \mathcal{A}^\ast | = 2^k \) for some \( k \in \mathbb{N} \). We construct a binary tree \( T \) with nodes corresponding to the subsets of \( \mathcal{A}^\ast \). In the root we put the entire set \( \mathcal{A}^\ast \). 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 \( s \) is conducted via the binary search in \( T \) starting at its root. Whenever the algorithm reaches the leaf, its corresponding action is assigned to the input state \( s \). Assume that the algorithm reached non-leaf node \( v \) of \( T \). Denote its children as: \( v_l \) and \( v_r \) and the corresponding action-sets as \( \mathcal{A}_{v_l}^\ast \) and \( \mathcal{A}_{v_r}^\ast \) respectively. For the ITT architecture, the following is true:

**Lemma 2.1.**

\[
P[\pi_\theta(s) \in \mathcal{A}_{v_l}^\ast | \pi_\theta(s) \in \mathcal{A}_{v_l}^\ast \cup \mathcal{A}_{v_r}^\ast] = \frac{\psi(\phi(l^\theta_{v_l}(s)))^\top \xi(v_l)}{\psi(\phi(l^\theta_{v_r}(s)))^\top (\xi(v_l) + \xi(v_r))},
\]

where \( \xi(v) \overset{\text{def}}{=} \sum_{a \in \mathcal{A}_v^\ast} \psi(\phi(l^\theta_a(s))) \).

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 \( a = \psi(\phi(l^\theta_{v_l}(s)))^\top \xi(v_l), b = \psi(\phi(l^\theta_{v_r}(s)))^\top \xi(v_r) \). 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 Updates: 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.

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, where to the best of our knowledge they were never tried before. Furthermore, as embarrassingly simple conceptually (yet very efficient at the same time), 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 $\theta = (\theta_1^T, \theta_2^T)^T \in \mathbb{R}^D$, 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 $\theta$ [Choromanski et al. (2018); Sutton and Barto (2018); Dou et al. (2022); Zhou et al. (2023); He et al. (2023c); Huang and Wang (2023)]. We conducted gradient-based optimization with the antithetic ES-gradient estimator applying orthogonal samples (see: Choromanski et al. (2018)), defined as follows:

$$\nabla M_{\text{ort}} F_\sigma(\theta) := \frac{1}{2\sigma M} \sum_{i=1}^{M} F^{\text{AT}(i)}$$

(10)

where $F^{\text{AT}(i)} := F(\theta + \sigma \varepsilon_i) \varepsilon_i - F(\theta - \sigma \varepsilon_i) \varepsilon_i$

for the hyper-parameter $\sigma > 0$ and where $(\varepsilon_i)_{i=1}^M$ have marginal distribution $\mathcal{N}(0, I_D)$, 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 $\theta$ as

$$\theta_{k+1} = \theta_k + \eta \nabla M_{\text{ort}} F_\sigma(\theta),$$

(11)

where the learning rate $\eta = 0.01$ is fixed throughout the experiments.

Theoretical Results on ES-optimization: We present in Appendix E theoretical analysis of the antithetic estimator, to shed light on its effectiveness (see Theorem E.4 and Lemma E.6). We will now focus on discussing our main experimental results.

4 EXPERIMENTS

We conducted three experiments: (a) compared ITT with IOT and explicit policies on 15 environments from OpenAI Gym and DeepMind Control Suite (DMCS) RL libraries [Brockman et al. (2016); Tassa et al. (2018), (b) compared regular ITT with its SRP-version, (c) compared regular ITT with its lazy-tower version (the latter two on the subset of these environments). We present neural network specifications and hyper-parameters in Appendix A.1. Our main findings are:

- (Section 4.1) Our ITT architecture achieves stronger performance than existing implicit policies and their explicit counterparts.
- (Section 4.2) Our ITT architecture can be used with sublinear time inner-product search algorithms, resolving the exponential sample complexity issue with existing implicit policies, with little sacrifice in performance.
- (Section 4.3) We shows that our ITT architecture allows for state and action towers to be updated at different rates, with little sacrifice in performance. Thus, we can achieve further computational gains.

4.1 ITTs Versus IOTs and Explicit Policies

For the OpenAI Gym environments, at each transition we were sampling a set of actions $A^* = \{a_1, ..., a_N\}$ (see Equation 11), and choosing an action according to Equation 3. For the DMCS environments, $A^*$ 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 $A^* = A$. For the environments with continuous actions, we set $N = 1000$ for the OpenAI Gym environments and $N = 10000$ for the DMCS environments. Figure 3 and Table I illustrate the results. 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, while having substantially faster iterations than IOTs (see:
Figure 3: The comparison of the performance of different policy-architectures: ITTs, IOTs and explicit on various OpenAI Gym and DeepMind Control Suite tasks. We plot average curves obtained from $s = 10$ seeds, and present the 90th percentile and the 10th percentile using shadowed regions. For fair comparison, we choose architectures’ sizes such that the number of trainable parameters of ITTs is comparable to but upper-bounded by that of IOTs and explicit variants (see: Appendix A with additional environments and experimental details).

Table 1: Setting analogous to the one from Fig. 3. 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.

| Task          | ITT       | IOT       | Explicit       |
|---------------|-----------|-----------|----------------|
| Swimmer6      | 988.3±10.1 | 984.9±28.1 | 767.4±120.2    |
| Swimmer15     | 995.8±7.0  | 990.7±0.001 | 936.0±48.7     |
| FishSwim      | 652.2±152.1 | 331.7±8.4  | 470.7±0.004    |

Wall-clock Time. Table 2 shows the wall-clock time (in hours on 24 CPU cores) for each policy-architecture for a selected set of environments. For each of these environments, we also include the number of training iterations. We conclude 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).

Compare to Additional Baselines. We provide comparison to previous works on different OpenAI Gym environments in Table 7 in Appendix A.3.
Implicit Two-Tower Policies

Figure 4: 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 90th percentile and the 10th percentile as shadowed regions. We present results on additional tasks in Appendix A.5.

Figure 5: 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 90th percentile and the 10th percentile using shadowed regions.

Table 2: Comparison of the wall-clock time (in hours) for different policy-architectures and a selected sample of the environments from Fig. 3. "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   |

4.2 Fast ITTs

ITT-SRP (signed random projections): The random projection vectors $\omega_1, ..., \omega_m$ have marginal distribution $N(0, I_d)$ 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 $b_i = \text{median}(\{\omega_i^\top \tilde{l}_j(a_j)\}_{j=1}^N)$ for each projection vector $\omega_i$, and map the action to the hashed space using $z \rightarrow (\text{sgn}(\omega_1^\top \tilde{z} - b_1), ..., \text{sgn}(\omega_m^\top \tilde{z} - b_m))^\top$. Figure 4 shows the comparison of the regular ITTs with ITTs applying SRPs. For HalfCheetah-v2, we use $N = 2^{14} = 16,384$ and $m = 6$. For Swimmer-v2, we use $N = 2^{10} = 1,024$ and $m = 3$. ITT-SRPs maintain good performance even though the hash-space is very small (of size $2^8 = 256$ for HalfCheetah-v2 and $2^7 = 128$ for Swimmer-v2). We chose $C = \max_{a \in A} \|l_\theta^a(a)\|_2$ (see: Eq. 6). We present results on additional tasks in Appendix A.5.

ITT-RFT (random feature trees): Similar to ITT-SRP, ITT-RFT could address exponential sample complexity issue of implicit policies, with little sacrifice in performance (see Appendix A.4).

ITT-lazy: we update the action tower every 5 iterations. Figure 5 shows the performance of this lazy variant is comparable to that of regular ITT, while the lazy action-tower variant achieves substantial wall-clock time savings (see Appendix A.6).

Ablation Studies: In Appendix B we provide extensive ablation studies, showing that (1) ITT consistently outperforms even when we significantly reduce the number of neurons; (2) ITT-SRP achieves substantial reduction in wall-clock time, especially when the number of samples required gets large; (3) ITT-SRP can be applied in either training or inference or in both.

Pair T-test Results: In Appendix C we provide
evidence of statistical significance that ITT achieves higher scores than IOT and explicit policies.

Compute resources. We use HP Enterprise XL170r server with 24 cpu cores and 128GB RAM.

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

Bartlett, P. L. (2002). An introduction to reinforcement learning theory: Value function methods. In Mendelson, S. and Smola, A. J., editors, *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, pages 184–202. Springer.

Bellman, R. (1954). Some applications of the theory of dynamic programming - A review. *Oper. Res.*, 2(3):275–288.

Bianchini, B., Halm, M., Matni, N., and Posa, M. (2021). Generalization bounds for implicit learning of nearly discontinuous functions. *arXiv preprint arXiv:2112.06881*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Choromanski, K., Chen, H., Lin, H., Ma, Y., Sehanobish, A., Jain, D., Ryoo, M. S., Varley, J., Zeng, A., Likhosherstov, V., Kalashnikov, D., Sindhwani, V., and Weller, A. (2021a). Hybrid random features. *ICLR 2022*, abs/2110.04367.

Choromanski, K., Pacchiano, A., Parker-Holder, J., Tang, Y., and Sindhwani, V. (2019). From complexity to simplicity: Adaptive es-active subspaces for blackbox optimization. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d’Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 10299–10309.

Choromanski, K., Rowland, M., Sindhwani, V., Turner, R. E., and Weller, A. (2018). Structured evolution with compact architectures for scalable policy optimization. In Dy, J. G. and Krause, A., editors, *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*, pages 969–977. PMLR.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlós, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. (2021b). Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021*, Virtual Event, Austria, May 3-7, 2021. OpenReview.net.

Choromanski, K. M., Rowland, M., and Weller, A. (2017). The unreasonable effectiveness of structured random orthogonal embeddings. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, December 4-9, 2017, Long Beach, CA, USA, pages 219–228.

Choromanski, K. M., Sehanobish, A., Lin, H., Zhao, Y., Berger, E., Parshakova, T., Pan, A., Watkins, D., Zhang, T., Likhosherstov, V., et al. (2023). Efficient graph field integrators meet point clouds. In *International Conference on Machine Learning*, pages 5978–6004. PMLR.

Dou, J. X., Pan, A. Q., Bao, R., Mao, H. H., and Luo, L. (2022). Sampling through the lens of sequential decision making. *arXiv preprint arXiv:2208.08056*.

Du, Y., Lin, T., and Mordatch, I. (2019). Model-based planning with energy-based models. In Kaelbling, L. P., Kragic, D., and Sugiura, K., editors, *3rd Annual Conference on Robot Learning, CoRL 2019*, Osaka, Japan, October 30 - November 1, 2019, *Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 374–383. PMLR.

Elmahachtoub, A., Gupta, V., and Zhao, Y. (2023a). Balanced off-policy evaluation for personalized pricing. In *International Conference on Artificial Intelligence and Statistics*, pages 10901–10917. PMLR.

Elmahachtoub, A. N., Lam, H., Zhang, H., and Zhao, Y. (2023b). Estimate-then-optimize versus integrated-estimation-optimization: A stochastic dominance perspective. *arXiv preprint arXiv:2304.06833*.

Florence, P., Lynch, C., Zeng, A., Ramirez, O. A., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. (2021). Implicit behavioral cloning. In Faust, A., Hsu, D., and Neumann, G., editors, *Conference on Robot Learning*, 8-11 Nove-
Implicit Two-Tower Policies

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. (2017). Reinforcement learning with deep energy-based policies. In Precup, D. and Teh, Y. W., editors, *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*, pages 1352–1361. PMLR.

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

He, S., Han, S., Su, S., Han, S., Zou, S., and Miao, F. (2023b). Robust multi-agent reinforcement learning with state uncertainty. *Transactions on Machine Learning Research*.

He, S., Wang, Y., Han, S., Zou, S., and Miao, F. (2023c). A robust and constrained multi-agent reinforcement learning framework for electric vehicle amod systems. *arXiv preprint arXiv:2209.08230*.

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

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

Huang, B. and Wang, J. (2023). 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*.

Huang, B., Zhao, T., Yue, M., and Wang, J. (2023). Bi-level adaptive storage expansion strategy for microgrids using deep reinforcement learning. *IEEE Transactions on Smart Grid*.

Jain, D., Caluwaerts, K., and Iscen, A. (2020). From pixels to legs: Hierarchical learning of quadruped locomotion. In Kober, J., Ramos, F., and Tomlin, C. J., editors, *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pages 91–102. PMLR.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., and Levine, S. (2018). Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *CoRR*, abs/1806.10293.

LeCun, Y., Chopra, S., Hadsell, R., Huang, F. J., and et al. (2006). A tutorial on energy-based learning. In *PREDICTING STRUCTURED DATA*. MIT Press.

Mania, H., Guy, A., and Recht, B. (2018). Simple random search of static linear policies is competitive for reinforcement learning. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2455–2467.

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

Neyshabur, B. and Srebro, N. (2015). On symmetric and asymmetric lshs for inner product search. In *International Conference on Machine Learning*, pages 1926–1934. PMLR.

Pham, N. (2021). Simple yet efficient algorithms for maximum inner product search via extreme order statistics. In Zhu, F., Ooi, B. C., and Miao, C., editors, *KDD ’21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 1339–1347. ACM.

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

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Gruber, L., Holzleitner, M., Adler, T., Kreil, D. P., Kopp, M. K., Klambauer, G., Brandstetter, J., and Hochreiter, S. (2021). Hopfield networks is all you need. In *International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Rawat, A. S., Chen, J., Yu, F. X., Suresh, A. T., and Kumar, S. (2019). Sampled softmax with random
fourier features. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d’Alché-Buc, F., Fox, E. B., and Garnett, R., editors, Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 13834–13844.

Salimans, T., Ho, J., Chen, X., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. CoRR, abs/1703.03864.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In International conference on machine learning, pages 1889–1897. PMLR.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. CoRR, abs/1707.06347.

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

Shrivastava, A. and Li, P. (2015a). Asymmetric minwise hashing for indexing binary inner products and set containment. In Gangemi, A., Leonardi, S., and Panconesi, A., editors, Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015, pages 981–991. ACM.

Shrivastava, A. and Li, P. (2015b). Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (MIPS). In Meila, M. and Heskes, T., editors, Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI 2015, July 12-16, 2015, Amsterdam, The Netherlands, pages 812–821. AUAI Press.

Singi, S., He, Z., Pan, A., Patel, S., Sigurdsson, G. A., Piramuthu, R., Song, S., and Ciocarlie, M. (2023). Decision making for human-in-the-loop robotic agents via uncertainty-aware reinforcement learning. arXiv preprint arXiv:2303.06710.

Song, Y. and Kingma, D. P. (2021). How to train your energy-based models. CoRR, abs/2101.03288.

Song, Z., Parr, R., and Carin, L. (2019). Revisiting the softmax bellman operator: New benefits and new perspective. In Chaudhuri, K. and Salakhutdinov, R., editors, 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, pages 5916–5925. PMLR.

Sundaram, N., Turmukhamedova, A., Satish, N., Mostak, T., Indyk, P., Madden, S., and Dubey, P. (2013). Streaming similarity search over one billion tweets using parallel locality-sensitive hashing. Proceedings of the VLDB Endowment, 6(14):1930–1941.

Sutton, R. S. (1998). Reinforcement learning: Past, present and future. In McKay, B., Yao, X., Newton, C. S., Kim, J., and Furuhashi, T., editors, 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, pages 195–197. Springer.

Sutton, R. S. and Barto, A. G. (1998). Reinforcement learning - an introduction. Adaptive computation and machine learning. MIT Press.

Sutton, R. S. and Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. (2018). Deepmind control suite. arXiv preprint arXiv:1801.00690.

van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In Schulmans, D. and Wellman, M. P., editors, Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA, pages 2094–2100. AAAI Press.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 5998–6008.

Watkins, C. J. C. H. and Dayan, P. (1992). Technical note q-learning. Mach. Learn., 8:279–292.

Xie, J., Lu, Y., Zhu, S.-C., and Wu, Y. (2016). A theory of generative convnet. In International Conference on Machine Learning, pages 2635–2644. PMLR.

Xu, Y., Xie, J., Zhao, T., Baker, C., Zhao, Y., and Wu, Y. N. (2022). Energy-based continuous inverse optimal control. IEEE transactions on neural networks and learning systems.

Yu, W., Jain, D., Escontrela, A., Iscen, A., Xu, P., Coumans, E., Ha, S., Tan, J., and Zhang, T. (2021). Visual- locomotion: Learning to walk on complex terrains with vision. In Faust, A., Hsu, D., and Neumann, G., editors, Conference on Robot Learning, 8-11 November 2021, London, UK, volume 164 of Proceedings of Machine Learning Research, pages 1291–1302. PMLR.
Zhou, H., Lan, T., and Aggarwal, V. (2022). Pac: Assisted value factorization with counterfactual predictions in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:15757–15769.

Zhou, H., Lan, T., and Aggarwal, V. (2023). Value functions factorization with latent state information sharing in decentralized multi-agent policy gradients. *IEEE Transactions on Emerging Topics in Computational Intelligence*. 

Appendix: Implicit Two-Tower Policies

A ADDITIONAL EXPERIMENTAL DETAILS

We include the code and instructions how to use it at https://anonymous.4open.science/r/itt-9881/README.md

A.1 Neural Network Specifications and Hyper-parameter Tuning

Neural Network Specifications Tables 4–5 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 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 layers and for all the methods (because in this environment only, using linear activation in all layers improves the performance of the baselines).

Hyperparameter Tuning Tables 3 illustrate fine-tuned hyper-parameter $\sigma$, 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 3: Fine-tuned hyper-parameter $\sigma$, used in ES gradient estimator calculations (Equation 10) for different environments and different policy-architectures.

A.2 ITT Base Variant Results on Additional Environments

In Table 6, we present final average scores together with their corresponding standard deviations (over $s = 10$ different seeds) for all 15 environments tested in the paper. In Figure 6, we provide plots showing training curves for additional OpenAI Gym tasks.

A.3 Compare to Additional Baselines

We provide comparison to previous works on different OpenAI Gym environments in Table 7. Note the ES baseline (Salimans et al., 2017) applies weights decay and performs fitness shaping, and they cap episode length at a constant steps for all workers, which they dynamically adjust during the training. Without applying any of these techniques used by the ES baseline to improve performance, and without fine-tuning parameters for the
Implicit Two-Tower Policies

| 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        |
| Acrobat-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:Swimmer6                      | 3100| 3120| 3100     |
| DMCS:HopperStand                   | 1980| 2000| 1980     |
| DMCS:WalkerWalk                    | 2200| 2220| 2200     |

Table 4: The dimensionality of the learnable $\theta \in \mathbb{R}^D$ for different environments and different policy-architectures.

| 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        |
| Acrobat-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 5: The number of layers of the neural networks encoding different architectures for different environments.

Figure 6: The comparison of the performance of different policy-architectures: ITTs, IOTs and explicit on additional OpenAI Gym tasks. We plot average curves obtained from $s = 10$ seeds, and present the $90^{th}$ percentile and the $10^{th}$ percentile using shadowed regions. For fair comparison, we choose architectures’ sizes such that the number of trainable parameters of ITTs is comparable to but upper-bounded by that of IOTs and explicit variants.

given reward thresholds (the number of ITT neural network layers and the value of $\sigma$ are the same as before), our
Table 6: Setting analogous to the one from Fig. 3. 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.

ITT architecture reaches given reward thresholds with less timesteps.

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

Table 7: 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).

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

A.4 ITT-RFT Results

In Table 8, we present results for ITT with random feature tree (ITT-RFT) described in Section 2.3.2. Random feature mechanisms are shown to provide substantial computational gains with little sacrifice in performance (Choromanski et al., 2023).

The results show that ITT-RFT achieves competitive performance in different environments, outperforming baselines. The result suggest that ITT-RFT could address the exponential sample complexity issue of implicit policies, without sacrificing performance.

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

Table 8: 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, described in Section 2.3.2.

A.5 ITT-SRP Results on Additional Environments

Figure 7 shows results of ITT-SRP on additional environments. For the additional environments, we use the number of action samples $N = 2^{10} = 1,024$ and the number of projection vectors $m = 3$. ITT-SRPs maintain good performance even though the hash-space is very small (of size $2^7 = 128$).
A.6 ITT with Lazy Action Updates

We record wall-clock time of ITT with lazy action updates on different OpenAI Gym environments. The results demonstrate that ITT with lazy action updates achieves substantial reduction in wall-clock time.

Table 9 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 9: 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).

B ABLATION STUDIES

B.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 10, 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 11, 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.
Table 10: We present the number of timesteps needed to reach given reward thresholds set by previous works \cite{Salimans2017, Schalman2015}. The results were averaged over 6 random seeds. “ES” stands for the results of explicit policies provided by \cite{Salimans2017}. “ITT-1” and “ITT-2” stand for variants of ITT with the number of neurons significantly reduced.

Table 11: The dimensionality of the learnable $\theta \in \mathbb{R}^D$ for different environments.

B.2 ITT-SRP Runtime Against the Sampling Amount

We record wall-clock time of ITT-SRP on different OpenAI Gym environments. The results demonstrate that ITT-SRP achieves substantial reduction in wall-clock time.

While running IOT, to ensure sufficient coverage of action space, we need to sample a large number of actions at each timestep (usually the number of samples is exponential in the dimension of the action space). To showcase the effectiveness of ITT-SRP in resolving the sample complexity of IOT, we require $N = 2^c$ action samples per timestep.

Table 12 shows the wall-clock time (in minutes on 30 CPU cores on Google Cloud Compute) for ITT-SRP and IOT. We conclude 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 on wall-clock times between ITT-SRP and IOT. 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-SRP is statistically significant at a 95% confidence level.

Table 12: 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 at a 95% confidence level (all p-values presented are below 0.05).
B.3 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 13 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          | ITT       | ITT       | ITT-SRP   | ITT-SRP   | # iter |
|-------------------|-----------|-----------|-----------|-----------|--------|
| Inference         |           |           |           |           |        |
| Swimmer-v2        | 344.13 ± 5.18 | 341.38 ± 7.93 | 333.84 ± 7.26 | 500       |
| LunarLanderC-v2   | 157.38 ± 71.81 | 154.41 ± 71.01 | 149.15 ± 69.27 | 500       |
| Hopper-v2         | 1007.95 ± 3.66 | 999.12 ± 11.39 | 1000.54 ± 7.74 | 500       |
| HalfCheetah-v2    | 2866.02 ± 416.81 | 2690.64 ± 139.05 | 2565.74 ± 143.15 | 4000     |

Table 13: 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.

C 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 14 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 14 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.

D Confidence Interval Plots

In the main paper (Figure 3), we presented average score and quantiles, which are key metrics commonly used. To further showcase the strength of proposed ITT architecture, below (Table 8) we present the scores in a similar fashion as in Figure 3 but using 95% confidence intervals instead of 90% and 10% percentiles.
| 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 14: 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 & Explicit p-value | ITT # iter | IOT # iter | Reward Threshold |
|----------------------|------------------|------------------------|------------|------------|-----------------|
| Swimmer-v2           | 2.96e-05         | 10.80                  | 420.10     | 67.98      | 70.98           |
| LunarLanderContinuous-v2 | 2.02e-07         | 12.20                  | 386.80     | -65.44     | -65.44          |
| MountainCar-v0       | 1.52e-07         | 733.80                 | 2000.00    | -180.0     | -180.0          |
| MountainCarContinuous-v0 | 6.38e-2          | 75.80                  | 125.00     | 23.36      | 23.36           |
| InvertedPendulumBulletEnv-v0 | 1.12e-07 | 9.50                   | 112.10     | 24.39      | 24.39           |

Table 15: 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.
Implicit Two-Tower Policies

Figure 8: The comparison of the performance of different policy-architectures: ITTs, IOTs and explicit on various OpenAI Gym tasks. We plot average curves and present the 95% confidence intervals using shadowed regions.

E Theoretical Results

E.1 Proof of results in Section 2

Proof of Lemma 2.1.

\[ P[\tau_0(s) \in \mathcal{A}^*_v | \tau_0(s) \in \mathcal{A}^*_v \cup \mathcal{A}^*_r] = \sum_{a \in \mathcal{A}^*_v} \exp(-E_\theta(s, a)) \]

\[ = \sum_{a \in \mathcal{A}^*_v \cup \mathcal{A}^*_r} \exp(-E(s, a)) \]

\[ = \sum_{a \in \mathcal{A}^*_v} \exp\left\{ K \left( \ell^a_S(s), \ell^a_A(a) \right) \right\} \]

\[ = \sum_{a \in \mathcal{A}^*_v \cup \mathcal{A}^*_r} \exp\left\{ K \left( \ell^a_S(s), \ell^a_A(a) \right) \right\} \]

\[ \approx \sum_{a \in \mathcal{A}^*_v \cup \mathcal{A}^*_r} \exp\left( \phi(\ell^a_S(s))^\top \phi(\ell^a_A(a)) \right) \]

\[ \approx \psi(\phi(\ell^a_S(s)))^\top \sum_{a \in \mathcal{A}^*_v} \psi(\phi(\ell^a_A(a))) \]

\[ \approx \psi(\phi(\ell^a_S(s)))^\top \psi(\phi(\ell^a_A(a))) \]

\[ = \psi(\phi(\ell^a_S(s)))^\top \left( \xi(v_l) + \xi(v_r) \right), \] (12)

\[ \Box \]

E.2 Proof of results in Section 3

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

Definition E.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

\[ \hat{\nabla}^{\text{AT}, \text{ort}} F_{\sigma}(\theta) := \frac{1}{2\sigma M} \sum_{i=1}^M F^{\text{AT}(i)} \text{ where } F^{\text{AT}(i)} := \Phi(\theta + \sigma \varepsilon_i) \varepsilon_i - \Phi(\theta - \sigma \varepsilon_i) \varepsilon_i \] (13)

\[ \hat{\nabla}^{\text{FD}, \text{ort}} F_{\sigma}(\theta) := \frac{1}{\sigma M} \sum_{i=1}^M F^{\text{FD}(i)} \text{ where } F^{\text{FD}(i)} := \Phi(\theta + \sigma \varepsilon_i) \varepsilon_i - \Phi(\theta) \varepsilon_i, \] (14)

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

Definition E.2 (Gaussian smoothing). The Gaussian smoothing of \(F(x)\) is defined as

\[ F_{\sigma}(\theta) = \frac{1}{\kappa} \int F(\theta + \sigma \varepsilon) e^{-\frac{1}{2}||\varepsilon||^2} d\varepsilon, \text{ where } \kappa = (2\pi)^{d/2}, \] (15)
and its gradient is
\[ \nabla F_\sigma(\theta) = \frac{1}{\sigma \kappa} \int F(\theta + \sigma \epsilon) e^{-\frac{1}{2} \|\epsilon\|^2_2} d\epsilon. \] (16)

**Assumption E.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
\[ F(\theta + \sigma \epsilon) = F(\theta) + \sigma \nabla F(\theta)^\top \epsilon + \frac{\sigma^2}{2} \epsilon^\top \nabla^2 F(\theta) \epsilon. \]

**Theorem E.4.** Suppose Assumption [E.3] holds. The mean squared error of the AT ES-gradient estimator applying orthogonal samples is
\[ \text{MSE} \left( \hat{\nabla}_M^{\text{AT,ort}} F_\sigma(\theta) \right) := \mathbb{E}\left[ \left\| \hat{\nabla}_N^{\text{AT,ort}} F_\sigma(\theta) - \nabla F_\sigma(\theta) \right\|^2_2 \right] = \frac{1}{M} \mathbb{E}\left[ \left\| \nabla F(\theta)^\top \epsilon \right\|_2^2 \right] - \left\| \nabla F_\sigma(\theta) \right\|^2_2. \]

The mean squared error of the FD ES-gradient estimator applying orthogonal samples is
\[ \text{MSE} \left( \hat{\nabla}_N^{\text{FD,ort}} F_\sigma(\theta) \right) := \mathbb{E}\left[ \left\| \hat{\nabla}_N^{\text{FD,ort}} F_\sigma(\theta) - \nabla F_\sigma(\theta) \right\|^2_2 \right] = \frac{1}{M} \mathbb{E}\left[ \left\| \nabla F(\theta)^\top \epsilon + \frac{\sigma^2}{2} \epsilon^\top \nabla^2 F(\theta) \epsilon \right\|_2^2 \right] - \left\| \nabla F_\sigma(\theta) \right\|^2_2. \]

**Remark E.5.** Theorem [E.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 and 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 and 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 E.6.** Assume \( M \leq D \), we have
\[ \text{MSE} \left( \hat{\nabla}_M^{\text{AT,ort}} F_\sigma(\theta) \right) = \frac{D + 2}{M} \left( \| \nabla F(\theta) \|^2_F - \| \nabla F_\sigma(\theta) \|^2_2 \right) \]
\[ + \frac{(D + 2)\sigma^4}{M} \left( \sum_{i=1}^D \nabla^2 F(\theta_{ii})^2 \right) - \| \nabla F_\sigma(\theta) \|^2_2, \]

and therefore
\[ \text{MSE} \left( \hat{\nabla}_M^{\text{FD,ort}} F_\sigma(\theta) \right) - \text{MSE} \left( \hat{\nabla}_M^{\text{AT,ort}} F_\sigma(\theta) \right) = \frac{(D + 4)\sigma^4}{4M} \| \nabla^2 F(\theta) \|^2_F + \frac{(D + 2)\sigma^4}{M} \left( \sum_{i=1}^D \nabla^2 F(\theta_{ii})^2 \right). \]

The proofs of Theorem [E.4] and Lemma [E.6] are at the end of Section [E].

**Remark E.7.** Suppose Assumption [E.3] holds. We observe that evaluating \( \nabla_M^{\text{AT,ort}} F_\sigma(\theta) \) requires \( 2M \) queries of \( F(\cdot) \) and evaluating \( \nabla_M^{\text{FD,ort}} F_\sigma(\theta) \) requires only \( M + 1 \) queries of \( F(\cdot) \). Consequently, FD-gradient estimator is preferred when \( \sigma^4 \| \nabla^2 F(\theta) \|^2_F \ll \| \nabla F(\theta) \|^2_2 \), and AT-gradient estimator is preferred when \( \sigma^4 \| \nabla^2 F(\theta) \|^2_F \gg \| \nabla F(\theta) \|^2_2 \), which is highly likely when \( \sigma \) is large.
proof of Theorem \[\text{E.4}\] AT ES-gradient estimator.

\[
\text{MSE} \left( \hat{N}^{AT,ort}_N F_\sigma(\theta) \right) = E \left[ \left\| \frac{1}{M} \sum_{i=1}^{M} F^{AT(i)} - \nabla F_\sigma(\theta) \right\|^2_2 \right]
\]

\[= E \left[ \left\| \frac{1}{M} \sum_{i=1}^{M} F^{AT(i)} \right\|^2_2 \right] - \left\| \nabla F_\sigma(\theta) \right\|^2_2
\]

The first term is

\[
E \left[ \left\| \frac{1}{M} \sum_{i=1}^{M} F^{AT(i)} \right\|^2_2 \right] = \frac{1}{M^2} \left( \sum_{i=1}^{M} E \left[ \left\| F^{AT(i)} \right\|^2_2 \right] + \sum_{i \neq j} E \left[ \langle F^{AT(i)}, F^{AT(j)} \rangle \right] \right)
\]

\[= \frac{1}{M^2} \left( \sum_{i=1}^{M} E \left[ \left\| F^{AT(i)} \right\|^2_2 \right] \right) = \frac{1}{M^2} \left( \sum_{i=1}^{M} E \left[ \frac{1}{2\sigma} (F(\theta + \sigma \varepsilon_i) - F(\theta) \varepsilon_i) \right] \right)
\]

\[= \frac{1}{M} E \left[ \left\| \frac{1}{2\sigma} (F(\theta + \sigma \varepsilon) \varepsilon - F(\theta) \varepsilon) \right\|^2_2 \right] = \frac{1}{M} E \left[ \left\| (\nabla F(\theta) \varepsilon) \varepsilon \right\|^2_2 \right],
\]

where the second equality is by orthogonality of \(\varepsilon_i\); the fourth equality is because \(\varepsilon_i\) are i.i.d.; the last equality is by Assumption \[\text{E.3}\]

**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.

\[
\frac{1}{M^2} \left( \sum_{i=1}^{M} E \left[ \left\| \tilde{F}^{FD(i)} \right\|^2_2 \right] \right) = \frac{1}{M^2} \left( \sum_{i=1}^{M} E \left[ \left\| \frac{1}{\sigma} (F(\theta + \sigma \varepsilon_i) - F(\theta) \varepsilon_i) \right\|^2_2 \right] \right)
\]

\[= \frac{1}{M} E \left[ \left\| \frac{1}{\sigma} (F(\theta + \sigma \varepsilon) \varepsilon - F(\theta) \varepsilon) \right\|^2_2 \right] = \frac{1}{M} E \left[ \left\| \varepsilon (\nabla F(\theta)^\top \varepsilon + \frac{\sigma^2}{2} \varepsilon^\top \nabla^2 F(\theta) \varepsilon) \right\|^2_2 \right],
\]

where the second equality is because \(\varepsilon_i\) are i.i.d., and the third equality is by Assumption \[\text{E.3}\]

**Proof of Lemma \[\text{E.6}\] AT ES-gradient estimator.**

\[
E \left[ \left\| (\nabla F(\theta)^\top \varepsilon) \varepsilon \right\|^2_2 \right] = \sum_{i,j,k} \nabla F(\theta)_i \nabla F(\theta)_j E \left[ \varepsilon_i \varepsilon_j \varepsilon_k^2 \right] = \sum_{i,k} \nabla F(\theta)_i^2 E \left[ \varepsilon_i^2 \varepsilon_k^2 \right]
\]

\[= \sum_{i=1}^{D} \nabla F(\theta)_i^2 E \left[ \varepsilon_i^4 \right] + \sum_{i=1}^{D} \nabla F(\theta)_i^2 \sum_{k \neq i} E \left[ \varepsilon_i^2 \varepsilon_k^2 \right] = (D + 2) \left\| \nabla F(\theta) \right\|^2_2,
\]

where the second equality is because odd moments of Gaussian r.v.s are zero. By Theorem \[\text{E.4}\] we have

\[
\text{MSE} \left( \hat{N}^{AT,ort}_M F_\sigma(\theta) \right) := E \left[ \left\| \hat{N}^{AT,ort}_N F_\sigma(\theta) - \nabla F_\sigma(\theta) \right\|^2_2 \right]
\]

\[= \frac{1}{M} E \left[ \left\| (\nabla F(\theta)^\top \varepsilon) \varepsilon \right\|^2_2 \right] - \left\| \nabla F_\sigma(\theta) \right\|^2_2 = \frac{D + 2}{M} \left\| \nabla F(\theta) \right\|^2_2 - \left\| \nabla F_\sigma(\theta) \right\|^2_2.
\]

**FD ES-gradient estimator.**
\[ E \left[ \left\| \varepsilon (\nabla F(\theta)^\top \varepsilon + \frac{\sigma^2}{2} \varepsilon^\top \nabla^2 F(\theta) \varepsilon) \right\|^2 \right] \]

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

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 \((D + 2) \left\| \nabla F(\theta) \right\|^2\) by the same argument as for the AT ES-gradient estimator; the third term is

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

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.