Importance of Representation Learning for Off-Policy Fitted Q-Evaluation

Xian Wu
University of California, Berkeley
xcwu@berkeley.edu

Nevena Lazic
DeepMind
nevena@google.com

Dong Yin
DeepMind
dongyin@google.com

Cosmin Paduraru
DeepMind
paduraru@google.com

Abstract

The goal of offline policy evaluation (OPE) is to evaluate target policies based on logged data with a possibly much different distribution. One of the most popular empirical approaches to OPE is fitted Q-evaluation (FQE). With linear function approximation, several works have found that FQE (and other OPE methods) exhibit exponential error amplification in the problem horizon, except under very strong assumptions. Given the empirical success of deep FQE, in this work we examine the effect of implicit regularization through deep architectures and loss functions on the divergence and performance of FQE. We find that divergence does occur with simple feed-forward architectures, but can be mitigated using various architectures and algorithmic techniques, such as ResNet architectures, learning a shared representation between multiple target policies, and hypermodels. Our results suggest interesting directions for future work, including analyzing the effect of architecture on stability of fixed-point updates which are ubiquitous in modern reinforcement learning.

1 Introduction

In many real-world applications of reinforcement learning (RL), policy optimization through direct (online) interaction with the environment is impractical due to constraints on safety, performance, or time. Environment access is often limited to a fixed offline (or batch) dataset of logged experience, collected by agents whose behavior is not controlled or even known. In such settings, one key challenge is *off-policy evaluation* (OPE): the task of estimating the value of a target policy based on batch data. Another related challenge is *offline policy selection*: the task of ranking a set of candidate policies based on their performance.

One of the most popular OPE methods is fitted Q-evaluation (FQE), an algorithm that tries to satisfy the Bellman equation of a target policy iteratively on batch data [Gordon, 1999]. From a theoretical perspective, Duan and Wang [2020] have shown that FQE is nearly minimax optimal under certain representational and distributional requirements, in particular that Q functions (action-value functions) are linear and Bellman backups applied to linear functions remain in the span of the features. While this *policy completeness* requirement used by Duan and Wang [2020] and other approaches [Chen and Jiang, 2019] is very strong, weaker assumptions such as linear realizability have recently been shown insufficient for reliable OPE [Wang et al., 2020a, Zanette, 2020]. It has also long been known that temporal-difference methods such as FQE are not even guaranteed to converge with linear function approximation and off-policy data [Tsitsiklis and Van Roy, 1997]. The same line of work establishes

Offline Reinforcement Learning Workshop at Neural Information Processing Systems, 2021.

that FQE can diverge with nonlinear approximation even with on-policy data. FQE can be stabilized through the use of regularization, but this may come at the expense of a large bias.

At the same time, on the empirical side, FQE has achieved promising results on a variety of benchmark OPE tasks [Fu et al., 2021, Paine et al., 2020] for which the assumptions required by theory do not hold. This suggests that many types of function approximators used in practice do not exhibit worst-case divergence behavior for FQE with off-policy data. Thus it is natural to consider beyond the worst case and investigate the effect of representations produced by modern deep learning methods on OPE performance, in particular, which types of common function approximators are stable and produce good estimates, and which types of popular deep neural networks are highly unstable with off-policy FQE. Initial negative results in this area were shown by Wang et al. [2021], who evaluated linear FQE with fixed pre-trained deep neural network representations produced by online RL, as well as with random Fourier features [Rahimi et al., 2007]. They found that error amplification does occur, except under very mild distribution shift.

In this work, we empirically examine the effect of *implicit regularization* on the performance and stability of off-policy FQE with rich function approximation. We demonstrate that certain choices of deep neural network architectures and losses can significantly improve the stability of FQE on strong benchmarks, and reduce or remove error amplification. In particular, we find that while off-policy FQE often diverges with linear and feed-forward architectures, this does not occur with ResNet [He et al., 2016] architectures, or when a shared representation or hypermodel Dwaracherla et al. [2020] is learned for evaluating multiple policies. Conditioned on no divergence, we observe the best approaches in terms of the performance metrics are not consistent across different environments and discount factors. Some approaches and architectures work well for different environments, and for other environments, the performance is comparable across techniques and architectures.

2 Preliminaries

Markov decision process. A Markov decision process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, r, P, \mu_0, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $P : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$ is the transition kernel, μ_0 is the initial state distribution, and γ is the discount. We assume that r, P, μ_0 are unknown. A policy $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$ is a mapping from a state to a distribution over actions. We use Π_π to denote the transition kernel from a state-action pair (s, a) to the next pair (s', a') under π. The expected return of a policy π is defined as

$$J_\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] \quad \text{where } s_0 \sim \mu_0,\ s_{t+1} \sim P(\cdot|s_t, a_t) \text{ and } a_t \sim \pi(\cdot|s_t)\,.$$

The value function $V_\pi(s)$ of a policy is the expected return conditioned on $s_0 = s$, and the action-value function Q_π is the expected return conditioned on $s_0 = s, a_0 = a$. They are the unique solutions of the Bellman equation

$$Q_\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} V_\pi(s').$$

We will use the shorthand notation $Q(s, \pi) = \mathbb{E}_{a \sim \pi(\cdot|s)}[Q(s, a)]$. Note that $V_\pi(s) = Q_\pi(s, \pi)$. Given the action-value function Q_π of a policy π and the initial state distribution μ_0, the expected return of the policy can be computed as

$$J_\pi = \mathbb{E}_{s \sim \mu_0}[Q_\pi(s, \pi)]. \tag{1}$$

Batch policy evaluation. In batch policy evaluation, we are given a *target* policy π, as well as a dataset $\mathcal{D}$ consisting of trajectories of (s, a, r, s') tuples generated by other policies, where $r = r(s, a)$ and $s' \sim P(\cdot|s, a)$. The goal is to evaluate the expected return J_π of a target policy. We will be interested in both estimating J_π and ranking a set of N candidate target policies $\pi_1, \ldots \pi_N$ according to their expected return.

Fitted Q-evaluation. Fitted Q-evaluation (FQE) is an algorithm that tries to estimate the action-value function Q_π of a policy π using iterative fixed-point updates corresponding to the Bellman equation. Given a dataset $\mathcal{D}$ of (s, a, r, s') transition tuples and a Q-function estimate parameterized by θ, the parameters are updated as

$$\theta_{i+1} = \arg\min_\theta \frac{1}{|\mathcal{D}|} \sum_{(s,a,r,s') \in \mathcal{D}} \left(r + \gamma Q(s', \pi(s')|\theta_i) - Q(s, a|\theta)\right)^2 .$$

One issue with FQE is that it may not converge with off-policy data even with linear value functions [Tsitsiklis and Van Roy, 1997].

3 Background and related work

3.1 OPE approaches

One set of approaches to batch policy evaluation are *value-based*: they try to estimate the value function of the target policy using either fixed point iteration or by minimizing losses related to the Bellman error. In addition to FQE, one such approach is Bellman residual minimization (BRM), where we minimize the squared Bellman error w.r.t. value function parameters rather than performing fixed-point updates. Unfortunately, this objective is well-known to be biased [Baird, 1995], where the bias is due to the variance of the empirical Bellman operator. Feng et al. [2019] and Uehara and Jiang [2020] minimize an alternative kernel-based loss, which yields a consistent estimate under certain conditions.

More recent OPE works rely on marginal importance sampling (MIS), and estimate the ratio between the occupancy distribution of the target policy and the data distribution. This approach was initiated by Liu et al. [2018]. Several recent MIS methods named DICE (distribution correction estimation) are based on the Lagrangian of the linear-programming formulation of the policy optimization problem [Nachum et al., 2019, Zhang et al., 2020, Yang et al., 2020, Dai et al., 2020], and as such involve saddle point optimization. Since the Lagrangian is not strongly-convex-strongly-concave even in the tabular case, it can be difficult to optimize, and typically additional regularizers are included in the objective [Yang et al., 2020]. The DICE estimates of the action-value function have empirically been observed to be less reliable than other methods [Fu et al., 2021].

Another class of methods for off-policy evaluation is *model-based*. These methods estimate an empirical model of the MDP (transitions and rewards) and then evaluate the target policy using this model. One approach to estimating dynamics is by training deep neural networks to maximize the log likelihood of the next state and reward given the current state and action, as in Zhang et al. [2021], Chua et al. [2018], Janner et al. [2019]. Other model-based methods may estimate dynamics over learned state representations, as in Schrittwieser et al. [2021].

See Levine et al. [2020] for a recent survey of offline reinforcement learning techniques.

3.2 FQE convergence and performance guarantees

An early analysis of convergence of temporal-difference (TD) methods in unichain MDPs was performed by Tsitsiklis and Van Roy [1997]. For the linear function approximation case, for a behavior policy β and target policy π, the iterative parameter updates can be written as a linear system $\theta^{(i+1)} = A_{\pi,\beta}\theta^{(i)} + b_\beta$. Here $A_{\pi,\beta}$ depends on the stationary distribution of β, as well as on π. When $\pi = \beta$, the spectral radius of $A_{\pi,\beta}$ can be shown to be less than 1, and the updates converge. When $\pi \neq \beta$, it is easy to construct examples for which the spectral radius is greater than 1 and the updates diverge. Divergence can be avoided through the use of regularization in the iterative least-squares updates, but the corresponding bias can be quite large.

More recently, Wang et al. [2020a] and Zanette [2020] have shown that even if we assume linear Q_π-realizability (where the Q-function of any policy is linear in known features) and that the batch data has good coverage, the worst-case sample complexity for OPE is exponential in the problem horizon. Wang et al. [2020a] show that sample-efficient OPE is possible under a much stronger representational condition of *policy completeness* [Duan and Wang, 2020], or under very mild distribution shift. Wang et al. [2021] provide similar results specifically for discounted linear FQE: the iterative linear updates can diverge (suffer geometric error amplification) in general, and that divergence is avoided under the same conditions of policy completeness or mild distribution shift.

The dynamics and convergence of fixed-point updates are considerably more difficult to analyze with non-linear function approximation. For certain non-linear function approximation, Tsitsiklis and Van Roy [1997] show that TD updates do not converge even with on-policy data. In this paper, we empirically examine the dynamics of off-policy FQE updates with two popular architectures: multilayer perceptron (MLP) and ResNet, as well as several variants to the iterative updates described next.

4 Policy evaluation techniques

Given a dataset $\mathcal{D}$ of transition tuples, a dataset $\mathcal{D}_0$ of initial states sampled from μ_0, and a set of policies $\{\pi_k\}_{k=1}^K$, we estimate the expected return J_{π_k} of each policy by estimating its action-value function Q_{π_k} and using Equation (1). We approximate action-value functions by deep neural networks, and consider two popular architectures: (1) multi-layer perceptron (MLP) with normalization at the input layer and ReLU activations, and (2) residual network (ResNet) [He et al., 2016], where each residual block is an MLP with a single hidden layer and input layer normalization. We consider the following learning approaches with both architectures:

- Vanilla FQE: we train each value function Q_{π_k} using standard FQE updates with the ℓ_2 loss.
- FQE with shared representation: to evaluate K target policies, we train a K-headed neural network where the k^{th} head corresponds to the action-value function Q_{π_k} of a policy π_k. More precisely, we combine a shared learned representation $\phi_\beta(s,a)$ with functions $f_{\theta_1}(\phi_\beta(s,a)), \ldots, f_{\theta_K}(\phi_\beta(s,a))$, where $\theta_1, \ldots \theta_K$ are policy-specific parameters. All parameters are trained simultaneously. The motivation for this multi-task learning approach is that it can yield better representations and improved generalization in supervised learning problems [Goodfellow et al., 2016]. Furthermore, another multi-task approach has recently been shown to improve the sample complexity of learning in linear MDPs with a generative model [Lu et al., 2021].
- FQE with a linear hypermodel: The hypermodel [Dwaracherla et al., 2020] trains an ensemble at scale and smooths the distribution of an ensemble of models initiated with different seeds. The hypermodel aims to smooth the PMF of the distribution of the estimates from discrete ensembles. More precisely, during evaluation, the linear hypermodel, in addition to taking as input the state-action pair, takes as input a seed value z drawn from a continuous distribution, in our case the 10-dimensional standard Gaussian. The base network corresponding to that seed value is $a^T z + b$, where $a \in \mathbb{R}^{10\times|\theta|}$ $b \in \mathbb{R}^{|\theta|\times 1}$ when the number of parameters for the base model is $|\theta|$. The learnable parameters for the hypermodel are a, b, so the goal is to find the a, b that minimizes the ℓ_2 loss for FQE for any randomly drawn z. The case of a discrete ensemble where the members are trained independently using different initial seeds, is equivalent to choosing z from the standard 10-dimensional basis vectors. Therefore, a hypermodel that takes in 10-dimensional Gaussian seeds can be interpreted as the smoothed version of an ensemble of 10 independently trained models. The hypermodel architecture implicitly regularizes the OPE estimates by introducing correlations between the model parameters via the distribution of z.
- Distributional FQE: we train each value function Q_{π_i} using the categorical distributional loss of Bellemare et al. [2017] with 41 atoms. This approach attempts to model the distribution of returns conditioned on a state-action pair (s,a), whereas $Q_\pi(s,a)$ simply corresponds to expected return. In the categorical version, this distribution is represented by a set of atoms $\{z_j\}_{j=1}^J$ that discretize possible return values, and probabilities corresponding to each atom $\{p_{\pi,j}(s,a)\}_{j=1}^J$. The value of a state can be computed as $Q_\pi(s,a) = \sum_{j=1}^J p_{\pi,j}(s,a) z_j$. Distributional FQE iteratively updates the atom probabilities using an approximation of the distributional Bellman error with a cross-entropy loss. One motivation for this approach is that by construction, the value function estimates cannot diverge outside of the range specified by the atoms. Thus, we only include this version of FQE for the purpose of comparing performance rather than divergence.

For all techniques and architectures described except for the hypermodel, we train an ensemble of 10 neural networks which are each initialized using a different random seed. Each member of the ensemble is also trained using a different bootstrapped subsampling of 50% of the episodes in the dataset. For the hypermodel, we use seeds drawn from the 10-dimensional Gaussian with unit variance and the randomness for the dataset bootstrapping for each training iteration is a function of the Gaussian seed used to train in that iteration. We choose 10-dimensional Gaussian seeds for the hypermodel because a hypermodel that takes in 10-dimensional Gaussian seeds can be interpreted as the smoothed version of an ensemble of 10 independently trained models.The main motivation for using ensembles (and their hypermodel approximation) is that this approach has recently yielded good empirical performance in bootstrapped off-policy approaches [Agarwal et al., 2019].

Implementation details. In all experiments, we use neural networks with hidden layer width of 1024 and 2, 3, or 4 layers / residual blocks. With shared representation, the last layer or block is task-specific, and all layers and blocks before the last are shared. We train all networks using the Adam [Kingma and Ba, 2014] optimizer with initial step size of 0.001.

5 Experiments

5.1 Setup

We evaluate the OPE approaches described in the previous section on a number of tasks from the DeepMind Control Suite [Tassa et al., 2018]. This a set of challenging continuous control tasks with simulators powered by the MuJoCo physics engine. We use benchmark datasets provided in [Gulcehre et al., 2020]. The datasets are generated by training policies online on each task until solved, and recording the data from the training run in episode format. The data generating policies are trained using D4PG [Barth-Maron et al., 2018], except for *manipulator:insert_ball* and *manipulator:insert_peg*, which used V-MPO [Song et al., 2019].

We evaluate the benchmark policies provided in Fu et al. [2021]. The policies are trained using four offline RL algorithms, D4PG [Barth-Maron et al., 2018], ABM [Siegel et al., 2020], CRR [Wang et al., 2020b], and behavior cloning. The policies produced by D4PG are deterministic, while the other policies are stochastic. For each of the 4 types of algorithms used to generate the policies, we choose the best and the worst policy based on their ground truth value based on the Monte Carlo simulations. This yields 8 policies per task. Because the policies are trained using algorithms that are regularized to be close to the data-generating policy, these best and worst policies span a large range of values in $[0, \frac{1}{1-\gamma}]$, which is the range of the values based on the effective horizon (all intermediate rewards for tasks in the DM Control Suite are bounded in $[0, 1]$). We estimate the values of the 8 policies per task (see Table 2 for policy identities), for discounts of $\gamma = 0.99$, $\gamma = 0.995$, and $\gamma = 0.999$, corresponding to horizons of different length.

For the methods that use ensembles (vanilla, shared representation, distributional), we train 10 different ensemble neural network functions to estimate the final Q function. We take 500 samples of state-action pairs from the initial state-action distribution of the environment and compute $Q(s, a)$ for each pair. Then we average over these values to compute the value according to this particular neural network function for a single policy. Thus, the final estimate $\hat{J}_\pi$ for each policy is the average of 10 value estimates. For the hypermodel, for each policy we train a single hypermodel and then take 100 samples from the hypermodel for each state-action pair. Since the hypermodel learns the pmf over the values, we take 100 samples to estimate the distribution[1]. We average over these 100 samples to determine $Q(s, a)$ for each s, a pair. Then we average over the 500 randomly drawn s, a pairs to get $\hat{J}_\pi$.

5.2 Results

Divergence. We first compare the MLP and ResNet architectures in terms of divergence. Table 1 shows the percentage of value estimates outside of the valid range $\left[0, \frac{1}{1-\gamma}\right]$ across all environments, policies, network depths, and ensemble members for the different methods (we omit distributional FQE since it always produces valid values by design). We observe that in nearly all cases, the ResNet architecture yields estimates of J_π that lie in the valid range. On the other hand, vanilla FQE with MLP architecture often produces values outside of the range, especially for high discount values. This is somewhat reduced by using hypermodels, and avoided with a shared representation. We also observe that shared representation is slightly worse compared to vanilla for ResNet, whereas it is the best approach to avoid divergence with MLP. Figures 2 and 3 show the progress of the value estimates using vanilla and hypermodel FQE using the MLP architecture with different numbers of layers as we perform more iterations of FQE. We see divergence for vanilla FQE and more stability with the hypermodel.

[1] It would take a sample complexity that is exponential in the dimension of the Gaussian seed (in our case 10) to fully estimate this pmf but here we use just 100 and we see that the performance is competetive to our other experiments. A deeper understanding of how to tune hypermodels is an interesting direction for research.

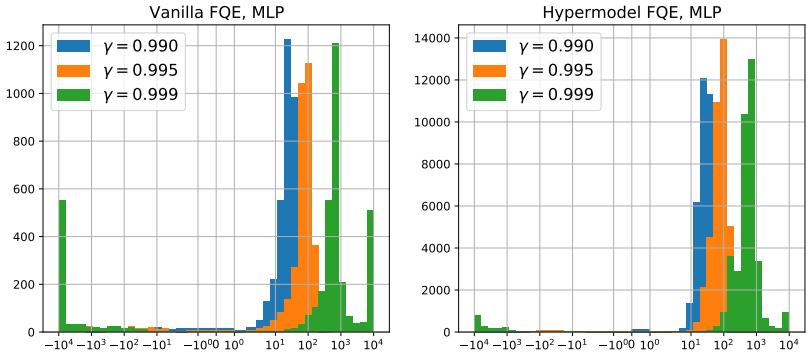

Figure 1: Histogram of value estimates $\widehat{J}_\pi$ obtained by running vanilla and hypermodel FQE with MLP architectures, across all environments, policies, and network depths. Very high value estimates indicate divergence.

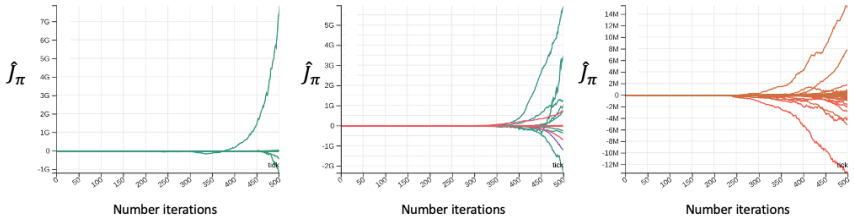

Figure 2: Value estimates during training run for several Finger:turn_hard policies at discount 0.999 using vanilla FQE with MLP architecture. The panel shows estimates for 3 different layer sizes: 4,3,2 hidden layers, respectively. Estimates diverge to infinity.

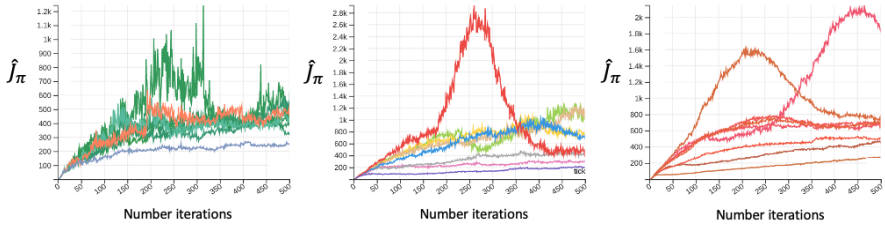

Figure 3: Value estimates during the training run for several Finger:turn_hard policies at discount 0.999 using hypermodel FQE with MLP architecture. The panel shows estimates for 3 different layer sizes: 4,3,2 hidden layers, respectively. Estimates are more stable, though sometimes out of range.

Figure 1 shows a histogram of the value estimates for vanilla FQE with the MLP architecture. The very high values of the estimates suggest that the algorithm is indeed diverging.

Table 1: Percentage of runs with value estimates outside of the valid range $\left[0, \frac{1}{1-\gamma}\right]$ across all environments, policies, network depths, and ensemble members.

	ResNet (%)			MLP (%)		
discount γ	vanilla	shared	hypermodel	vanilla	shared	hypermodel
0.990	0.10	0.15	0.0	6.27	0.41	0.77
0.995	0.10	0.13	0.0	19.1	0.49	3.46
0.999	0.08	0.05	0.0	40.5	0.9	14.5

Performance. We observe that the best approaches in terms of the performance metrics are not consistent across different environments and discount factors. We compare the different approaches in terms of the following metrics proposed in Fu et al. [2021]: (1) mean absolute error, (2) rank correlation between the estimator's ranking and the ranking according to ground truth, and (3) regret@1, the difference between the true value of the best policy and the true value of the predicted best policy. To compute the metrics, we predict the value of each policy as the mean of the in-range values predicted by ensemble members. For the few cases where all ensemble members predict values outside of $[0, \frac{1}{1-\gamma}]$, we set the predicted value to 0. We estimate the ground truth values using Monte Carlo simulation. The results for each environment, FQE version, and architecture are shown in figures 5 and 6 (absolute error), 7 and 8 (rank correlation), and 9 and 10 (regret@1) in the Appendix. We show one figure for regret@1 in Figure 4, which shows that a well-tuned method can identify the best policy during evaluation. Overall, conditioned on no divergence, we find that there is no overwhelmingly favorable technique or tuning that works well for a broad range of environments.

6 Discussion

FQE is one of the most popular approaches to off-policy evaluation, and has achieved promising empirical results on a variety of OPE benchmarks [Paine et al., 2020, Fu et al., 2021]. From a theoretical standpoint, several previous works have analyzed the dynamics of linear off-policy FQE, and found that it diverges except under the very strong assumption of policy completeness, or under minimal distribution shift [Tsitsiklis and Van Roy, 1997, Wang et al., 2020a, 2021]. When policy completeness holds, FQE is known to be nearly minimax optimal [Duan and Wang, 2020], but this assumption is unlikely to be true in practice.

In this work, we have performed an empirical investigation of the effects of *implicit regularization* on the dynamics of deep off-policy FQE. We have found that FQE often diverges with MLP network architectures (especially for high discounts γ), and that divergence does not occur with ResNet architectures. We have also found that with MLPs, divergence can be reduced when networks are trained using hypermodels, and mitigated by learning a shared representation for multiple policies. In terms of performance, conditioned on no divergence, we found that there is no clear winner between different approaches (and even for different depths within the same approach), and the performance largely depends on the task and environment.

Based on our results, we conclude that it is advantageous for applications of FQE to experiment with a hypermodel or shared representation along with the ResNet architecture, instead of the simple MLP even though MLP is a very popular and simple architecture. One interesting direction for future work is to analyze at a deeper level why these techniques are inherently more stable than MLPs.

References

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. Striving for simplicity in off-policy deep reinforcement learning. 2019.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.

Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR, 2017.

Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *International Conference on Machine Learning*, pages 1042–1051. PMLR, 2019.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114*, 2018.

Bo Dai, Ofir Nachum, Yinlam Chow, Lihong Li, Csaba Szepesvári, and Dale Schuurmans. Coindice: Off-policy confidence interval estimation. *arXiv preprint arXiv:2010.11652*, 2020.

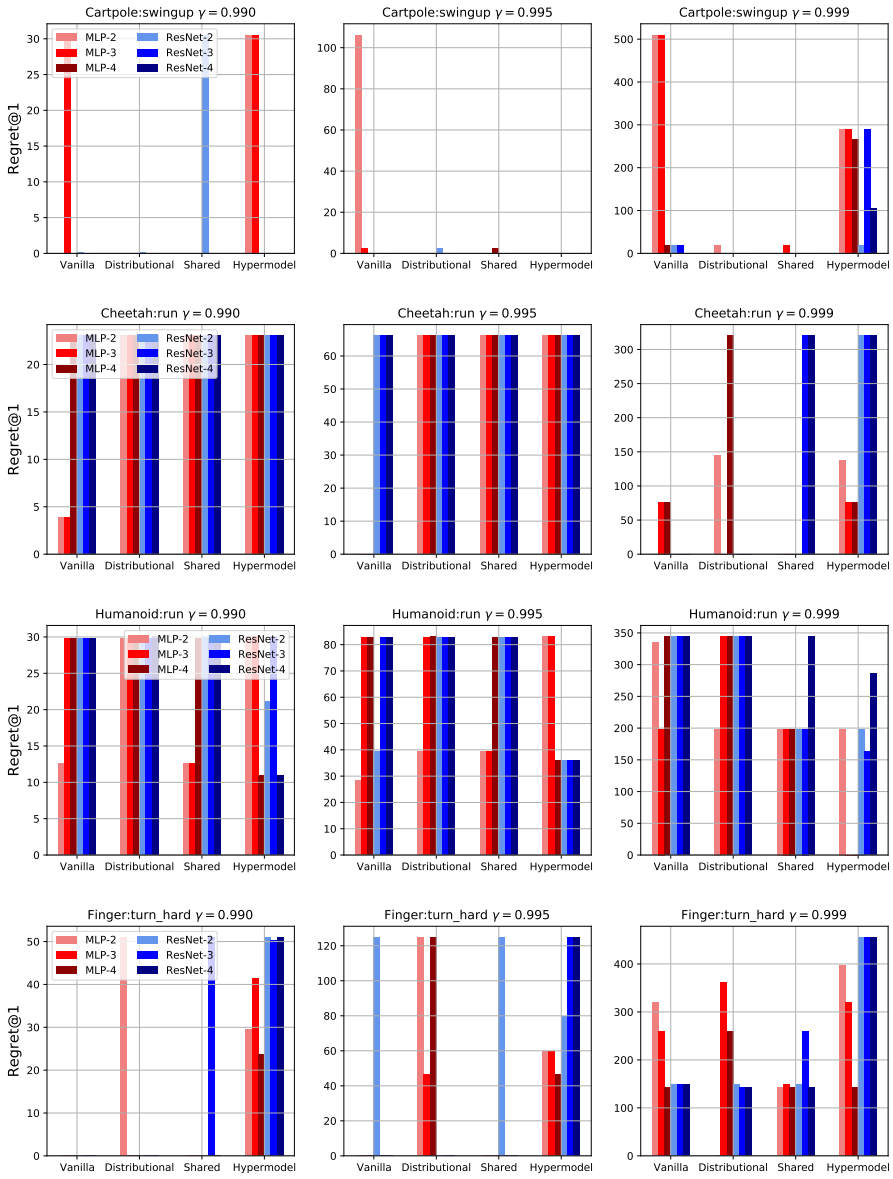

Figure 4: Regret@1 for the Cartpole:swingup, Cheetah:run, Humanoid:run, and Finger:turn_hard environments, for four different versions of FQE and MLP and ResNet architectures of depth 2, 3, and 4. For some environments like Cartpole and Finger, the methods explored in this paper will identify a very good policy (so regret@1 is 0).

Yaqi Duan and Mengdi Wang. Minimax-optimal off-policy evaluation with linear function approximation. *arXiv preprint arXiv:2002.09516*, 2020.

Vikranth Dwaracherla, Xiuyuan Lu, Morteza Ibrahimi, Ian Osband, Zheng Wen, and Benjamin Van Roy. Hypermodels for exploration. *arXiv preprint arXiv:2006.07464*, 2020.

Yihao Feng, Lihong Li, and Qiang Liu. A Kernel Loss for Solving the Bellman Equation. In *Advances in Neural Information Processing Systems 32*, pages 15456–15467. Curran Associates, Inc., 2019.

Justin Fu, Mohammad Norouzi, Ofir Nachum, George Tucker, Ziyu Wang, Alexander Novikov, Mengjiao Yang, Michael R Zhang, Yutian Chen, Aviral Kumar, et al. Benchmarks for deep

off-policy evaluation. *arXiv preprint arXiv:2103.16596*, 2021.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Geoffrey J Gordon. *Approximate solutions to Markov decision processes*. Carnegie Mellon University, 1999.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gomez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *arXiv preprint arXiv:2006.13888*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *arXiv preprint arXiv:1906.08253*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems*, pages 5356–5366, 2018.

Rui Lu, Gao Huang, and Simon S Du. On the power of multitask representation learning in linear mdp. *arXiv preprint arXiv:2106.08053*, 2021.

Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. DualDICE: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems*, pages 2315–2325, 2019.

Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter selection for offline reinforcement learning. *arXiv preprint arXiv:2007.09055*, 2020.

Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, volume 3, page 5. Citeseer, 2007.

Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. Online and offline reinforcement learning by planning with a learned model. *arXiv preprint arXiv:2104.06294*, 2021.

Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

H Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruva Tirumala, et al. V-mpo: On-policy maximum a posteriori policy optimization for discrete and continuous control. *arXiv preprint arXiv:1909.12238*, 2019.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

John N Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE transactions on automatic control*, 42(5):674–690, 1997.

Masatoshi Uehara and Nan Jiang. Minimax weight and q-function learning for off-policy evaluation. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Ruosong Wang, Dean P Foster, and Sham M Kakade. What are the statistical limits of offline rl with linear function approximation? *arXiv preprint arXiv:2010.11895*, 2020a.

Ruosong Wang, Yifan Wu, Ruslan Salakhutdinov, and Sham M. Kakade. Instabilities of offline RL with pre-trained neural representation. *CoRR*, abs/2103.04947, 2021. URL https://arxiv.org/abs/2103.04947.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Jost Tobias Springenberg, Scott Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *arXiv preprint arXiv:2006.15134*, 2020b.

Mengjiao Yang, Ofir Nachum, Bo Dai, Lihong Li, and Dale Schuurmans. Off-policy evaluation via the regularized lagrangian. *arXiv preprint arXiv:2007.03438*, 2020.

Andrea Zanette. Exponential lower bounds for batch reinforcement learning: Batch rl can be exponentially harder than online rl. *arXiv preprint arXiv:2012.08005*, 2020.

Michael R Zhang, Tom Le Paine, Ofir Nachum, Cosmin Paduraru, George Tucker, Ziyu Wang, and Mohammad Norouzi. Autoregressive dynamics models for offline policy evaluation and optimization. *arXiv preprint arXiv:2104.13877*, 2021.

Ruiyi Zhang, Bo Dai, Lihong Li, and Dale Schuurmans. GenDICE: Generalized offline estimation of stationary values. *arXiv preprint arXiv:2002.09072*, 2020.

A Additional experimental details and results

Table 2: Identities of policies evaluated for each environment.

Environment	Policy ids
Cartpole:swingup	16, 718, 514, 299, 721, 515, 300, 10
Cheetah:run	42, 334, 747, 540, 322, 30, 742, 544
Humanoid:run	110, 119, 698, 280, 496, 279, 709, 485
Finger:turn_hard	64, 665, 653, 455, 222, 233, 440, 52
Fish:swim	80, 783, 82, 355, 556, 767, 553, 357
Manipulator:insert_ball	137, 376, 805, 584, 141, 799, 380, 579
Manipulator:insert_peg	164, 158, 472, 248, 680, 674, 243, 467
Walker:walk	205, 839, 838, 638, 431, 640, 424, 194

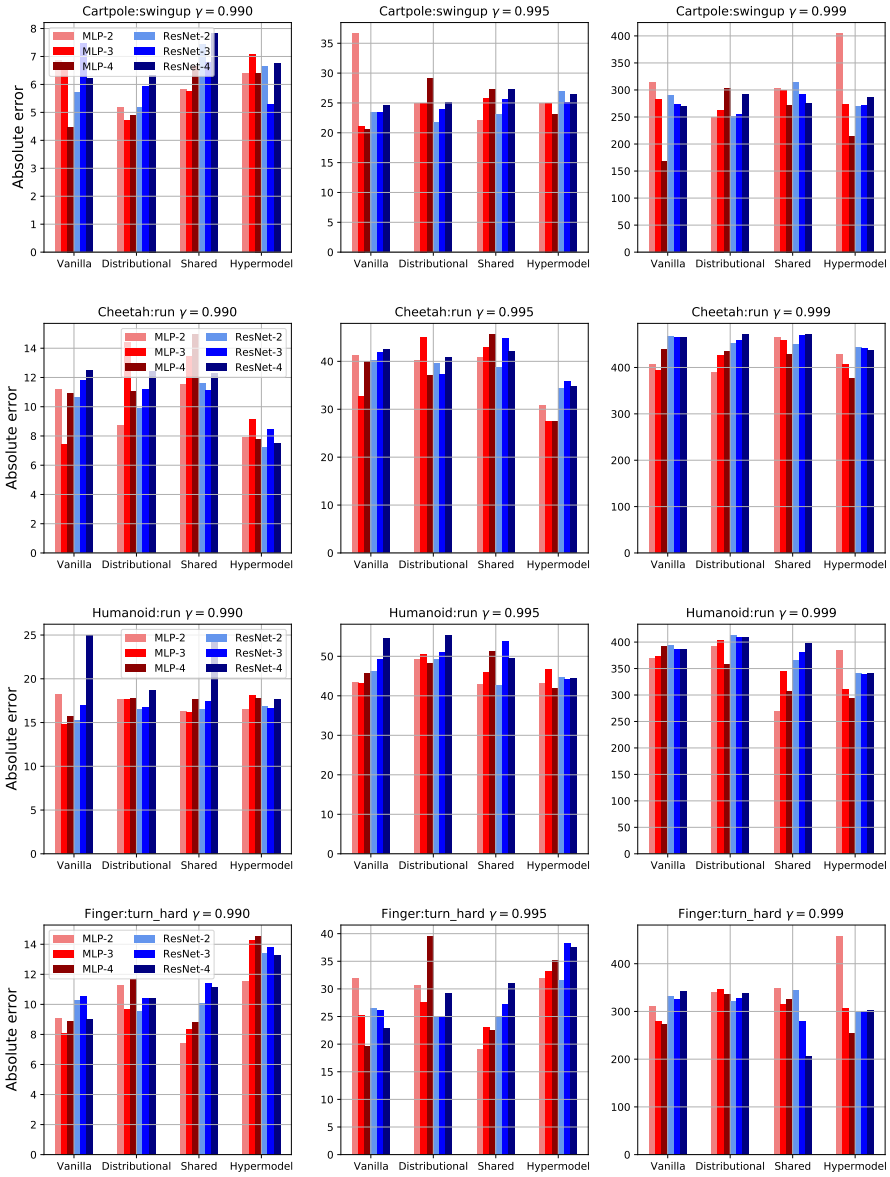

Figure 5: Mean absolute error in predicting policy values for the Cartpole:swingup, Cheetah:run, Humanoid:run, and Finger:turn_hard environments, for four different versions of FQE and MLP and ResNet architectures of depth 2, 3, and 4. Most of the results are comparable. For specific environments, spikes in RMSE for ResNet-based models vs MLP-based models can be observed.

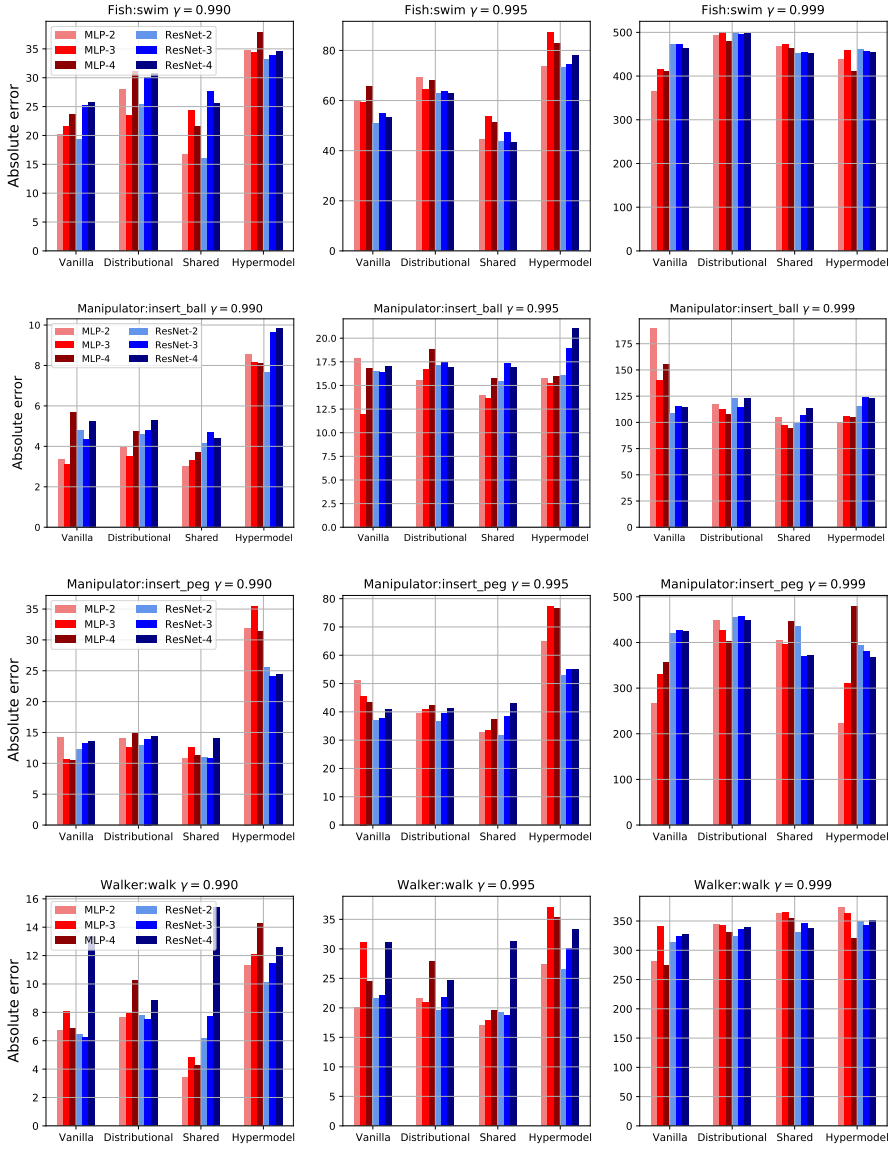

Figure 6: Mean absolute error in predicting policy values for the Fish:swim, Manipulator:insert_ball, Manipulator:insert_peg, and Walker:walk environments, for four different versions of FQE and MLP and ResNet architectures of depth 2, 3, and 4. Results are largely comparable. For some environments and discounts (ie Manipulator), hypermodels perform worse than other techniques.

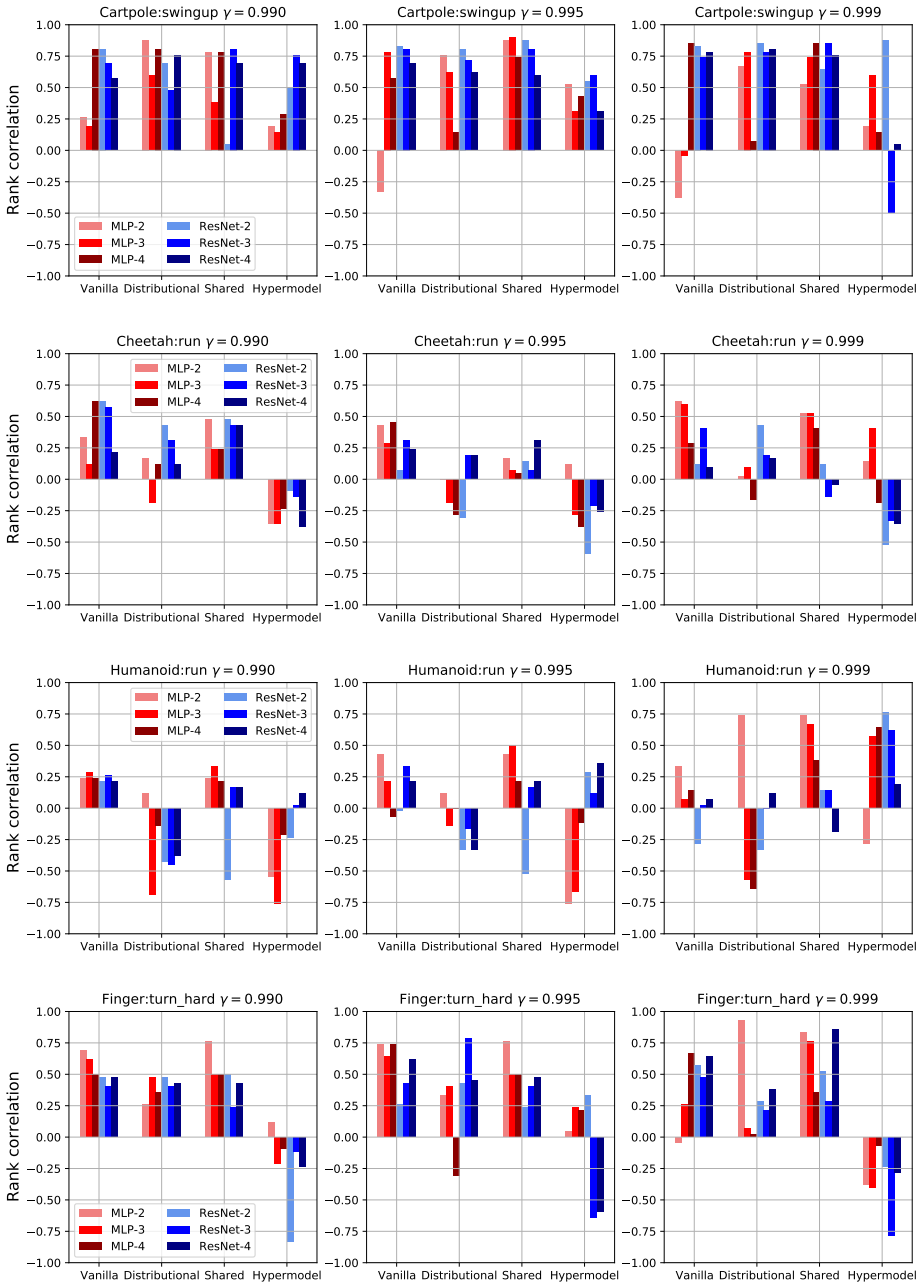

Figure 7: Rank correlation of true and predicted policy values for the Cartpole:swingup, Cheetah:run, Humanoid:run, and Finger:turn_hard environments, for four different versions of FQE and MLP and ResNet architectures of depth 2, 3, and 4. The hypermodels are not as competitive as the other methods, and for the humanoid environment, distributional FQE also performs poorly.

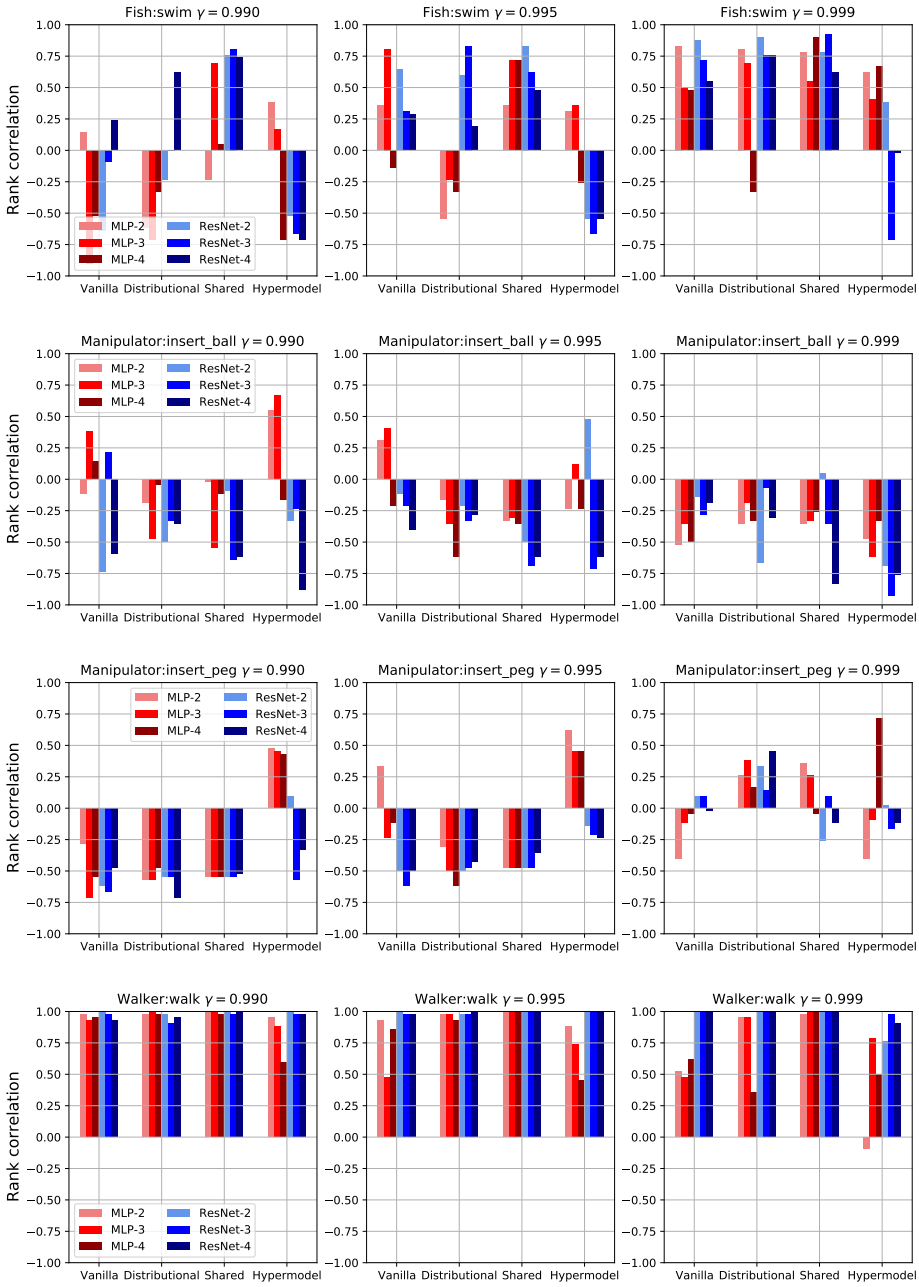

Figure 8: Rank correlation of true and predicted policy values for the Fish:swim, Manipulator:insert_ball, Manipulator:insert_peg, and Walker:walk environments, for four different versions of FQE and MLP and ResNet architectures of depth 2, 3, and 4. The Manipulator environments seem to be hard for every algorithm, while Walker seems easier.

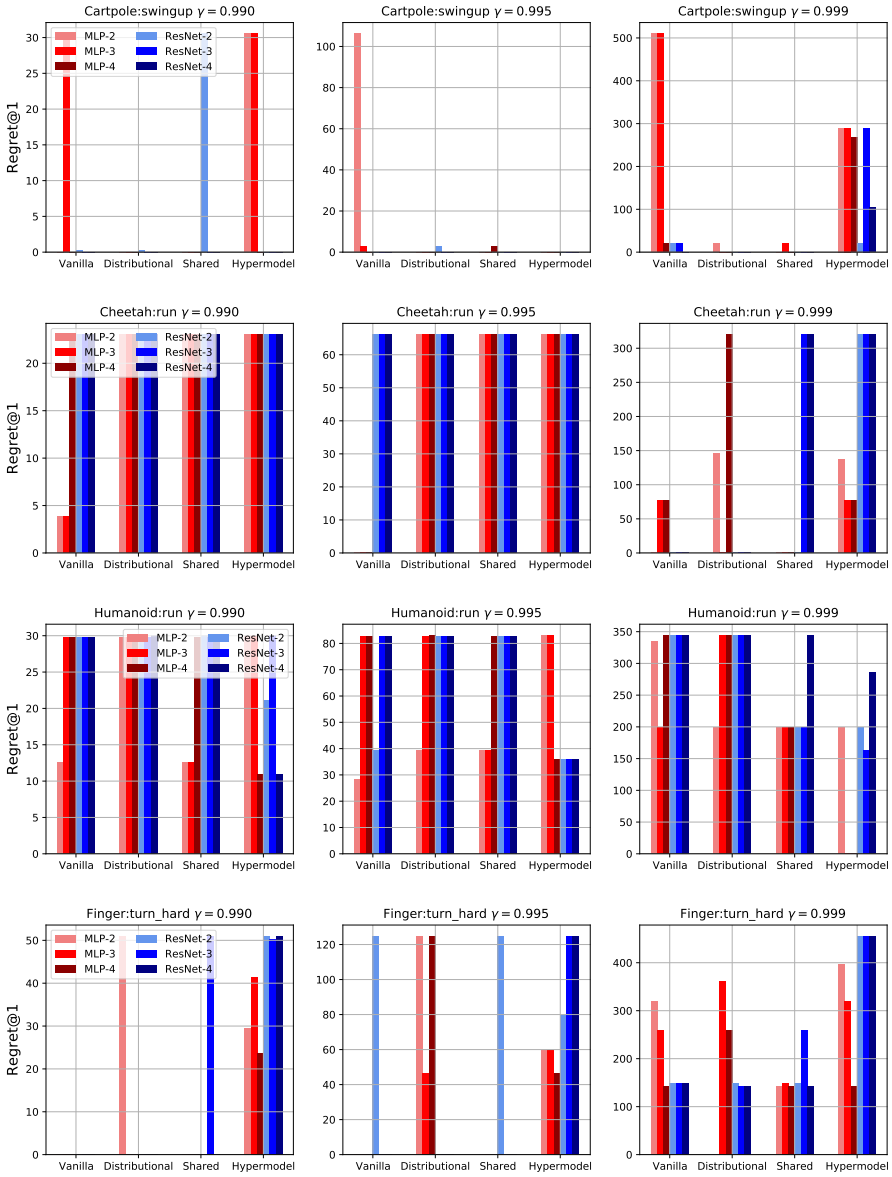

Figure 9: Regret@1 for the Cartpole:swingup, Cheetah:run, Humanoid:run, and Finger:turn_hard environments, for four different versions of FQE and MLP and ResNet architectures of depth 2, 3, and 4. For some environments like Cartpole and Finger, the methods explored in this paper will identify a very good policy (so regret@1 is 0).

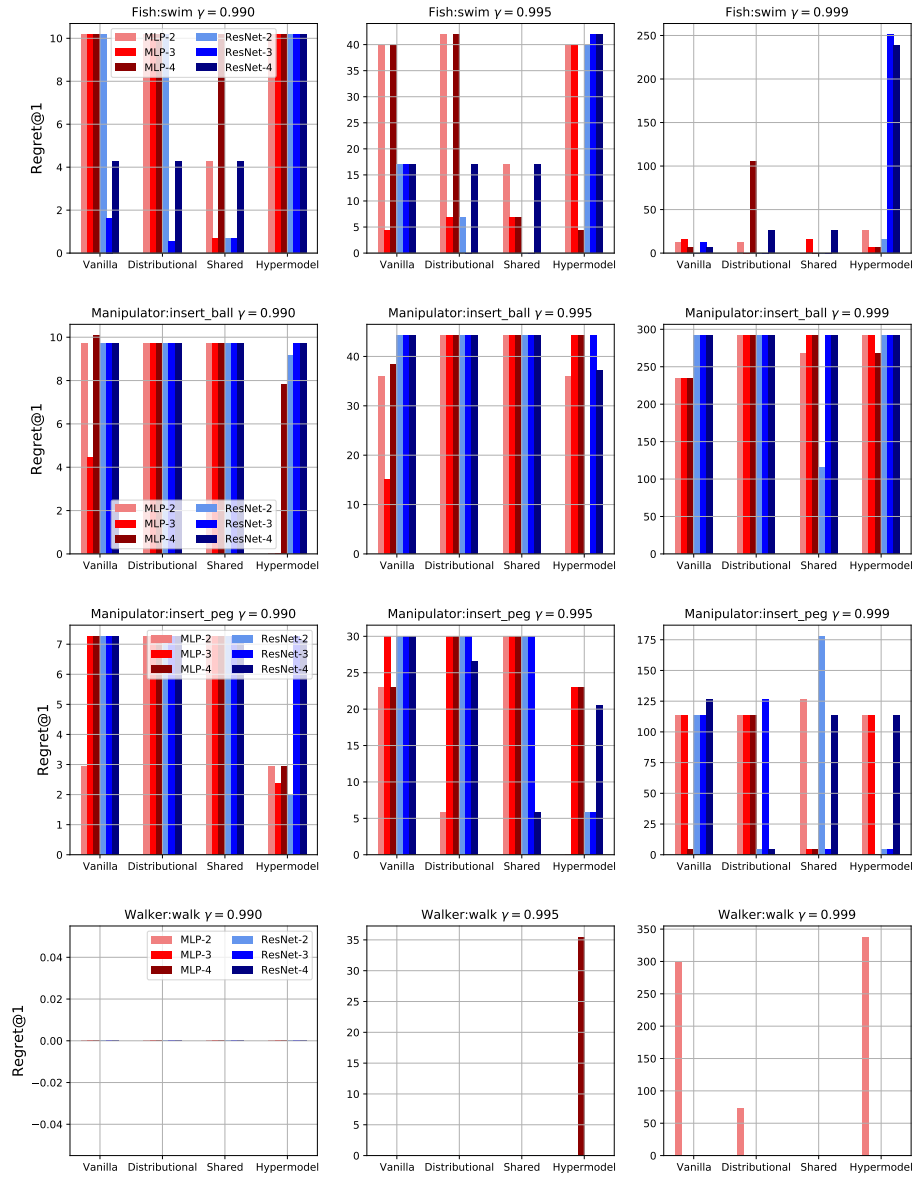

Figure 10: Regret@1 for the Fish:swim, Manipulator:insert_ball, Manipulator:insert_peg, and Walker:walk environments, for four different versions of FQE and MLP and ResNet architectures of depth 2, 3, and 4. Again we see that the Manipulator environments are very difficult to solve for our techniques. Walker is very simple; many techniques correctly identified the best policy (so regret@1 is 0).