Understanding and Preventing Capacity Loss in Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

The reinforcement learning (RL) problem is rife with sources of non-stationarity that can destabilize or inhibit learning progress. We identify a key mechanism by which this occurs in agents using neural networks as function approximators: *capacity loss*, whereby networks trained to predict a sequence of target values lose their ability to quickly fit new functions over time. We demonstrate that capacity loss occurs in a broad range of RL agents and environments, and is particularly damaging to learning progress in sparse-reward tasks. We then present a simple regularizer, Initial Feature Regularization (InFeR), that mitigates this phenomenon by regressing a subspace of features towards its value at initialization, improving performance over a state-of-the-art model-free algorithm in the Atari 2600 suite. Finally, we study how this regularization affects different notions of capacity and evaluate other mechanisms by which it may improve performance.

1 Introduction

Deep reinforcement learning has achieved remarkable successes in a variety of environments (Mnih et al., 2015; Moravčík et al., 2017; Silver et al., 2017; Abreu et al., 2019), but the precise reasons for its successes largely remain mysterious. Existing algorithms are highly sensitive to hyperparameters and seemingly innocuous design choices, to the extent that even minor variations to state-of-the-art methods can fail to make learning progress on tasks originally solved with ease. These instabilities are particularly pronounced in sparse-reward environments, where even different random seeds of the same algorithm can attain dramatically different performance outcomes. In these settings, even if the agent has experienced the rewards necessary to learn a high-scoring policy, it will often fail to translate those rewards into successful policy improvements. This presents a stark contrast to supervised learning, where existing approaches are reasonably robust to small hyperparameter changes, random seeds, and GPU parallelisation libraries. Moreover, naive applications of supervised learning methods such as momentum-based optimization or data augmentation to the RL problem have produced mixed results, and often require additional modification to yield performance improvements (Bengio et al., 2020; Raileanu et al., 2020).

We hypothesize that the non-stationary prediction problems agents face in RL may be a driving force in the challenges described above. RL agents must solve a sequence of similar prediction problems as they iteratively improve their prediction accuracy and their policy (Dabney et al., 2021), and solving each subproblem in this sequence is necessary to progress to the next subproblem. As shown in work on active learning, warm-starting a network by fitting similar data to that used in the downstream task can hurt the network's final accuracy, even if this data is drawn from the same distribution (Ash & Adams, 2020). This suggests that the slowly-shifting input and target distributions RL agents face may be particularly ill-suited to function approximation by deep neural networks. Indeed, several prior works studying the effect of re-initializing network parameters in reinforcement learning have found this enables agents to break through plateaus and improve generalization performance (Igl et al., 2021; Fedus et al., 2020).

The principal thesis of this paper is that over the course of training, deep RL agents lose their capacity to quickly fit new prediction tasks, and in extreme cases this capacity loss prevents the agent entirely from making learning progress. We present a rigorous empirical analysis of this phenomenon which considers both the ability of networks to learn new target functions via gradient-based optimization methods, and their ability to linearly disentangle states' feature representations.

We confirm that agents' ability to fit new target functions declines over the course of training in environments from the Atari suite (Bellemare et al., 2013) and non-stationary reward prediction tasks. We further find that the ability of representations to linearly distinguish different states, a proxy for their ability to represent certain functions, quickly diminishes in sparse-reward games, leading to representation collapse, where the feature outputs for every state in the environment inhabit a low-dimensional – or possibly even zero – subspace. Finally, we show evidence that representation collapse is a key factor in agents' failure to make learning progress in sparse-reward environments. We then propose a simple regularization technique, Initial Feature Regularization (InFeR), to prevent representation collapse: regress a set of feature projection heads to their values at initialization. This method improves performance on a number of sparse-reward environments and also increases the measures of capacity we are interested in, preventing representation collapse in sparse-reward environments and improving target-fitting capacity in dense-reward environments.

One striking take-away from our results is that agents trained on so-called 'hard exploration' games such as Montezuma's Revenge can attain significant improvements over existing competitive baselines *without* using smart exploration algorithms. This suggests that the poor performance of deep RL agents in sparse-reward environments is not solely due to inadequate exploration, but rather also in part due to poor representation learning. Essentially, agents which are 'too good' at predicting the zero value function lose their ability to fit the non-zero targets necessary for policy improvement. We believe this has significant implications for how the community views the interplay of exploration and representation learning in sparse reward environments.

2 Background

We consider the reinforcement learning problem wherein an agent seeks to maximize expected return in an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, R, \mathcal{P}, \gamma)$, where $\mathcal{X}$ denotes the state space, $\mathcal{A}$ the action space, R the reward function, $\mathcal{P}$ the transition probability function, and γ the discount factor. We will be primarily interested in *value-based* RL, where the objective is to learn the value function $Q^\pi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ associated with some (possibly stochastic) policy $\pi : \mathcal{X} \to \mathscr{P}(\mathcal{A})$, defined as $Q^\pi(x, a) = \mathbb{E}_{\pi,\mathcal{P}}[\sum_{k=0}^{\infty} \gamma^k R(x_k, a_k)|x_0 = x, a_0 = a]$. In particular, we are interested in learning the value function associated with the optimal policy π^* which maximizes the expected discounted sum of rewards from any state.

In Q-Learning (Watkins & Dayan, 1992), the agent performs updates to minimize the distance between a predicted action-value function Q and the bootstrap target defined as

$$\mathcal{T}Q(x, a) = \mathbb{E}[R(x_0, a_0) + \gamma \max_{a' \in \mathcal{A}} Q(x_1, a')|x_0 = x, a_0 = a] \; . \tag{1}$$

In most practical settings, updates are performed with respect to sampled transitions rather than on the entire state space. The target can be computed for a sampled transition (x_t, a_t, r_t, x_{t+1}) as $\hat{\mathcal{T}}Q(x_t, a_t) = r_t + \gamma \max_a Q(x_{t+1}, a)$.

When a deep neural network is used as a function approximator (the deep RL setting), Q is defined to be the output of a neural network with parameters θ, and updates are performed by gradient descent on sampled transitions $\tau = (x_t, a_t, r_t, x_{t+1})$. A number of tricks are often used to improve stability: the sample-based objective is minimized following stochastic gradient descent based on minibatches sampled from a replay buffer of stored transitions, and a separate set of parameters $\bar{\theta}$ is used to compute the targets $Q_{\bar{\theta}}(x_{t+1}, a_{t+1})$ which is typically updated more slowly than the network's online parameters. This yields the following loss function (where τ denotes the sampled transition):

$$\ell_{TD}(Q_\theta, \tau) = (R_{t+1} + \gamma \max_{a'} Q_{\bar{\theta}}(X_{t+1}, a') - Q_\theta(X_t, A_t))^2 \; . \tag{2}$$

In this work we will be interested in how additional representation-learning objectives shape agents' learning dynamics. We will refer to the outputs of the final hidden layer of the network (i.e. the penultimate layer) as *features*, denoted $\phi_\theta(x)$. Our choice of the penultimate layer is motivated by prior literature studying representations in RL (Ghosh & Bellemare, 2020; Kumar et al., 2021), although many works studying representation learning consider the outputs of earlier layers as well. In general, the features of a neural network are defined to be the outputs of whatever layer is used to compute additional representation learning objectives.

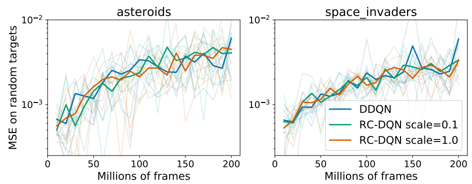

Figure 1: Neural networks exhibit a decline in their ability to fit random network outputs over the course of training in two demonstrative Atari environments.

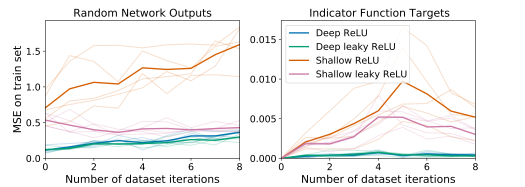

Figure 2: Deeper analysis on a sequential MNIST prediction setting showing that target-fitting capacity sees the greatest reduction in low capacity networks with ReLU units.

3 Capacity Loss

In this section we show that neural networks progressively lose their ability to quickly fit new targets when trained on sequential prediction tasks (i.e. settings in which the agent must solve a regression problem that iteratively changes over the course of training) including but not limited to those found in value-based RL. We find that capacity loss is particularly pronounced in sparse prediction tasks, where many of the target values the agent seeks to predict are zero. To study the effect of extreme capacity loss on performance in greater depth, we present a special case of the target-fitting capacity measure which is efficient to compute and has the intuitive interpretation of measuring the ability of the representation to linearly disentangle states. We find evidence that agents which have greater capacity according to this metric tend to achieve better performance in challenging environments from the Atari suite where agents fail to match human performance, and that those suffering from representation collapse according to this metric fail to make any learning progress at all.

3.1 Target-fitting capacity

Neural networks trained with temporal difference learning objectives must solve a sequence of target prediction problems. Ideally, the network should be able to fit later targets as quickly and accurately as it is able to fit early targets. However, prior work suggests that using 'warm starts', i.e. weights from a network trained on a different dataset or target function, can slow down training on new data points (Ash & Adams, 2020). We are therefore interested in studying whether this notion of interference may affect the ability of reinforcement learning agents to make learning progress. We begin by formally defining a notion of capacity loss which measures whether the network can quickly fit new targets, for example after a policy improvement step increases the expected reward obtained in a given state or after the target network is updated.

Definition 1 (Target-fitting capacity)**.** *Let $P_X \in \mathscr{P}(X)$ be some distribution over inputs X and $P_\mathcal{F}$ a distribution over a family of functions $\mathcal{F}$ with domain X. Let $\mathcal{N} = (g_\theta, \theta_0)$ represent the pairing of a neural network architecture with some initial parameters θ_0, and $\mathcal{O}$ correspond to an optimization algorithm for supervised learning. We measure the capacity of $\mathcal{N}$ under the optimizer $\mathcal{O}$ to fit the data-generating distribution $\mathcal{D} = (P_X, P_\mathcal{F})$ as follows:*

$$\mathcal{C}(\mathcal{N}, \mathcal{O}, \mathcal{D}) = \mathbb{E}_{f \sim P_\mathcal{F}}[\mathbb{E}_{x \sim P_X}[(g_{\theta'}(x) - f(x))^2]] \quad \textit{where } \theta' = \mathcal{O}(\theta_0, P_X, f)\,. \tag{3}$$

This definition of capacity takes into account the interaction between the optimization algorithm and the initial parameters, and can be adapted to a broad range of function classes such as the network's TD targets on the current dataset, the reward function on the environment, or randomly generated functions. The choice of function class is crucial to the notion of capacity measured. Using bootstrap targets from the network's current parameters can identify networks which cannot fit their current target function, but may reveal more about the complexity of the network output than the network's capacity to fit novel targets. A network which can only output the zero function, for example, will attain very low Bellman error in a sparse-reward environment, but may not be able to fit more interesting value functions. In our evaluations, we will therefore consider a random class of functions which is independent of the current network parameters.

To evaluate target-fitting capacity in deep RL agents, we sample target functions by randomly initializing neural networks with new parameters, and use the outputs of these networks as targets for regression. We then load initial parameters from an agent checkpoint at some time t, and regress on random network outputs over a fixed set of inputs sampled from the earliest available checkpoint's replay buffer. This ensures that the same regression problem is being evaluated at every checkpoint. We then evaluate the mean squared error after training for fifty thousand steps. We observe that as training progresses agents' checkpoints on average get modestly worse at fitting these random targets in most environments; due to space limitations we only show two representative environments where this phenomenon occurs in Figure 1, and defer the full evaluation to Appendix B. While the resulting curves are relatively noisy, we do see consistent results across both the double DQN agent and the agent trained with an additional auxiliary loss. We now turn our attention to a less computationally expensive experimental setting where it is easier to obtain statistically robust results.

We are particularly interested in understanding *why* capacity loss occurs. Two possible causes are immediate: the effect of *bootstrapping*, and the effect of *sequential training*. The effect of bootstrapping on capacity has been studied in other contexts (Mobahi et al., 2020; Kumar et al., 2021). We aim to isolate the effect of sequential prediction tasks on capacity loss. To minimize the potential for confounding factors to influence our results, we construct a toy prediction problem on the MNIST data set. We first consider labels computed by a randomly initialized neural network f_θ: we transform input-label pairs (x, y) from the canonical MNIST dataset to $(x, f_\theta(x))$, where $f_\theta(x)$ is the network output. To generate a new task, we simply reinitialize the network; our evaluations consist of 10 such iterations. We further consider a 'sparse-reward' version of MNIST: for each of 10 iterations i, we use the label $\hat{y}_i = \mathbb{1}[y < i]$. This mimics sparse-reward environments where the agent initially obtains no reward in the environment, then gradually improves its policy and thus increases its prediction targets over the course of training.

In Figure 2 we see that the networks trained in these two experiments both exhibit decreased ability to fit later target functions under a fixed optimization budget. This effect is strongest in small networks with ReLU activations, suggesting that some units may be saturating, but we see a similar trend across most architectures and prediction tasks. The sparse reward setting is particularly intriguing: we do not expect to see a monotone increase in error as the later label functions correspond to 'easier' learning problems (i.e. predicting the majority class will already yield reasonably low prediction error), but we do see that for equal difficulty, the network obtains greater error on the later target set than the earlier one, and this effect is significantly more pronounced than in the random labels tasks. This suggests that sparse reward signals can be particularly damaging to the ability of networks to fit new target functions.

3.2 Representation collapse and performance

Having shown that networks do indeed lose some notion of target-fitting capacity, we now turn our attention to the interaction between capacity and performance. To study this phenomenon further, we will use a computationally cheaper measure of network capacity which we call the effective dimension. An approximation of the rank of a feature embedding, it is both more computationally efficient – as it does not require training a network for several thousand steps – and requires fewer hyperparameter design choices than the previous capacity metric. While this does not take into account the ability of the network to update early layers, Appendix B shows that it correlates reasonably well with target-fitting capacity while also being a cheap, low-variance estimator against which to compare an agent's relatively noisy performance.

Definition 2 (Effective Dimension)**.** *Let $\phi : \mathcal{X} \to \mathbb{R}^d$ be a feature mapping. Let $\mathbf{X}_n \in \mathcal{X}^n$ be a tuple of n states in $\mathcal{X}$ sampled from some fixed distribution P. Fix $\varepsilon \geq 0$, and let $\phi(\mathbf{X}_n) \in \mathbb{R}^{n \times d}$ denote the matrix whose rows are the feature embeddings of states $x \in \mathbf{X}_n$. Let $\mathrm{SVD}(M)$ denote the multiset of singular values of a matrix M. Then the effective dimension of ϕ given input distribution P is defined to be*

$$\rho(\phi, P, \epsilon) = \lim_{n \to \infty} \mathbb{E}_{\mathbf{X}_n \sim P}[|\{\sigma \in \mathrm{SVD}\left(\frac{1}{\sqrt{n}}\phi(\mathbf{X}_n)\right) | \sigma > \varepsilon\}|] \tag{4}$$

for which a consistent estimator can be constructed as follows

$$\hat{\rho}_n(\phi, \mathbf{X}, \epsilon) = |\{\sigma \in \mathrm{SVD}\left(\frac{1}{\sqrt{n}}\phi(\mathbf{X})\right) | \sigma > \varepsilon\}| \, . \tag{5}$$

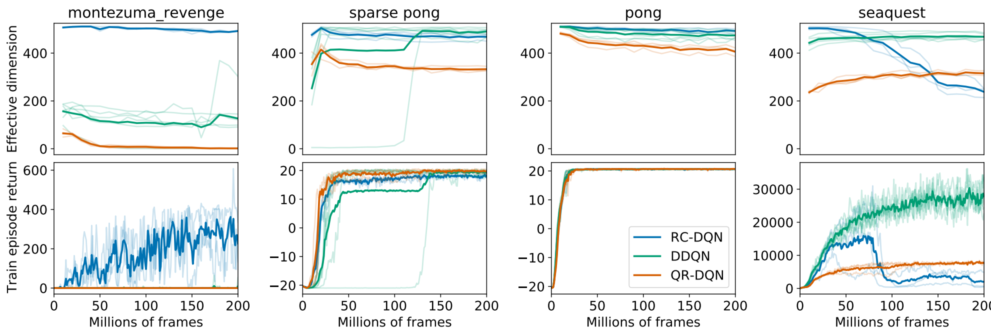

Figure 3: Effective dimension (top) and performance (bottom) over the course of training. We observe that effective dimension is higher for environments and auxiliary tasks which provide denser reward signals than for sparse reward problems.

The effective dimension is equal to the dimension of the subspace spanned by $\phi(\mathcal{X}) = \{\phi(x) \mid x \in \mathcal{X}\}$ when $\varepsilon = 0$ and the state space $\mathcal{X}$ is finite. For $\epsilon > 0$, it throws away small components of the feature embedding. In the language of Definition 1, this has the interpretation of measuring the ability of a linear classifier with bounded weight norm to distinguish the feature embeddings of input states. We show that ρ is well-defined and that $\hat{\rho}_n$ is a consistent estimator in Appendix A. Our analysis in this section superficially resembles that of Kumar et al. (2021), but with two key differences: first, we consider a different definition of capacity motivated by our previous discussion; concretely, our estimator does not normalize by the maximal singular value. Second, we are interested in the capacity of agents with unlimited opportunity to interact with the environment. Online data collection allows for interaction between the representation and behaviour policy, an important feature missing from the limited-data and offline settings. Thus prior observations on implicit under-parameterization in the data-limited regime do not immediately explain our observations on capacity loss, nor do they imply that effective dimension should decline in the online RL setting.

In our empirical evaluations, we train a double DQN (DDQN) agent, a quantile regression (QRDQN) agent, and a double DQN agent with an auxiliary random cumulant prediction task (RCDQN) (Dabney et al., 2021), on environments from the Atari suite, then evaluate $\hat{\rho}_n$ on agent checkpoints obtained during training. We train on 4 environments: Montezuma's Revenge (sparse reward), Pong (dense reward), a sparsified version of Pong in which the agent does not receive negative reward when its opponent scores, and Seaquest (dense reward, but more challenging than Pong). We run 3 random seeds on each environment-agent combination.

We visualize agents' effective dimension and performance in Figure 3. Two trends are immediately clear: at the environment level, agents tend to have higher effective dimension in environments with more reward signal. At the learning objective level, incorporating auxiliary rewards prevents representations from collapsing and in most settings increases the effective dimension over the double DQN objective, but can have a detrimental effect on learning progress in complex, dense-reward games presumably due to interference between the random rewards and the true learning objective. Intriguingly, over the course of training we do not see a consistent downward trend across all games and update objectives, suggesting that while effective dimension is indeed effective at identifying representation collapse it does not perfectly correlate with network capacity.

We observe that representation collapse is most pronounced in the QR-DQN objective, which in sparse-reward environments requires all 201 quantile heads to predict the zero target function. As a result, the agent's representation predictably collapses after training in Montezuma's Revenge to a low-dimensional subspace, in some cases converging to an effective dimension of exactly zero. We observe a similar, but less pronounced, phenomenon in the double DQN agent.

While the many moving parts in deep RL algorithms make it difficult to isolate the causal effect of a single representation property on performance, Figure 4a reveals a correlation between learning progress and effective dimension on challenging games where agents fail to achieve human-level performance. We see this correlation in both a Rainbow (Hessel et al., 2018) agent, and an agent trained with the regularizer we will introduce in the coming section – its precise form is not relevant

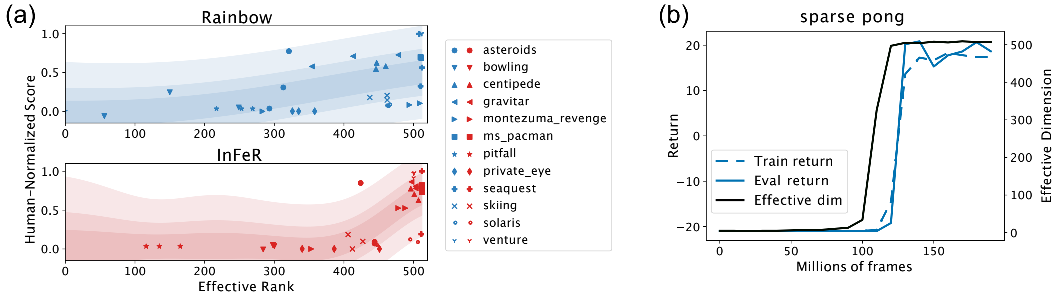

Figure 4: **(a)**: Agent capacity vs human-normalized score in games where Rainbow does not achieve superhuman performance. While effective dimension does not appear to solely determine agent performance, there is a positive correlation between effective dimension and human-normalized score. **(b)** An 'unlucky' seed from our evaluations on the sparsified version of Pong, where learning progress occurs only after the agent recovers from representation collapse.

to the discussion of feature collapse. Further, it is clear that agents whose representations have collapsed do not make learning progress, as demonstrated by the learning curves shown in Figure 4b, which highlights a particularly unlucky random seed that did not observe a point being scored during its initial random exploration period and experienced representation collapse. Eventually, after several million training frames, its effective dimension increased dramatically, and shortly *after* this occurred, the agent solved the task. We conclude that in its extreme form, representation collapse can completely prevent learning progress, but that the relationship between learning progress and effective dimension in less extreme cases is complex and one of many factors influencing performance in RL.

4 InFeR: Mitigating Capacity Loss with Feature Regularization

The previous section showed that capacity loss occurs in deep RL agents trained with online data, and in some cases appears to be a bottleneck to performance. We now consider how it might be mitigated, and whether explicitly regularizing the network to preserve its initial capacity improves performance in environments where representation collapse occurs. Our approach involves a function-space perspective on regularization, encouraging networks to preserve their ability to output linear functions of their features at initialization.

4.1 Feature-space regularization

Our goal is to train a network which preserves its ability to quickly fit new target functions. To do this, we propose using a training objective that preserves the network's ability to represent functions it had the capacity to output when it was initialized. Much like parameter regularization schemes which seek to keep *parameters* close to their initial values, we wish to regularize a set of network *outputs* towards their initial values. Similar perspectives have been used to prevent catastrophic forgetting in continual learning (Benjamin et al., 2019). There are many approaches that one could take to achieve this goal; the one we will present, Initial Feature Regularization (InFeR), applies an ℓ_2 regularization penalty on the output-space level by regressing a set of auxiliary linear projection heads on top of the feature layer of the network to match their outputs at initialization.

In our approach, illustrated in Figure 5, we begin with a fixed deep Q-learning neural network with parameters θ, and modify the network architecture by adding k auxiliary linear prediction heads g_i on top of the feature representation ϕ_θ. We take a snapshot of the agent's parameters at initialization θ_0, and use the outputs of the k auxiliary heads under these parameters as auxiliary prediction targets. We then construct an auxiliary loss computing the mean squared error between the outputs of the heads under the current parameters $g_i(x;\theta_t)$ and their outputs at initialization $g_i(x;\theta_0)$. This approach has the interpretation of amplifying and preserving subspaces of the features that were present at initialization. In practice, we found that scaling the auxiliary head outputs by a constant β

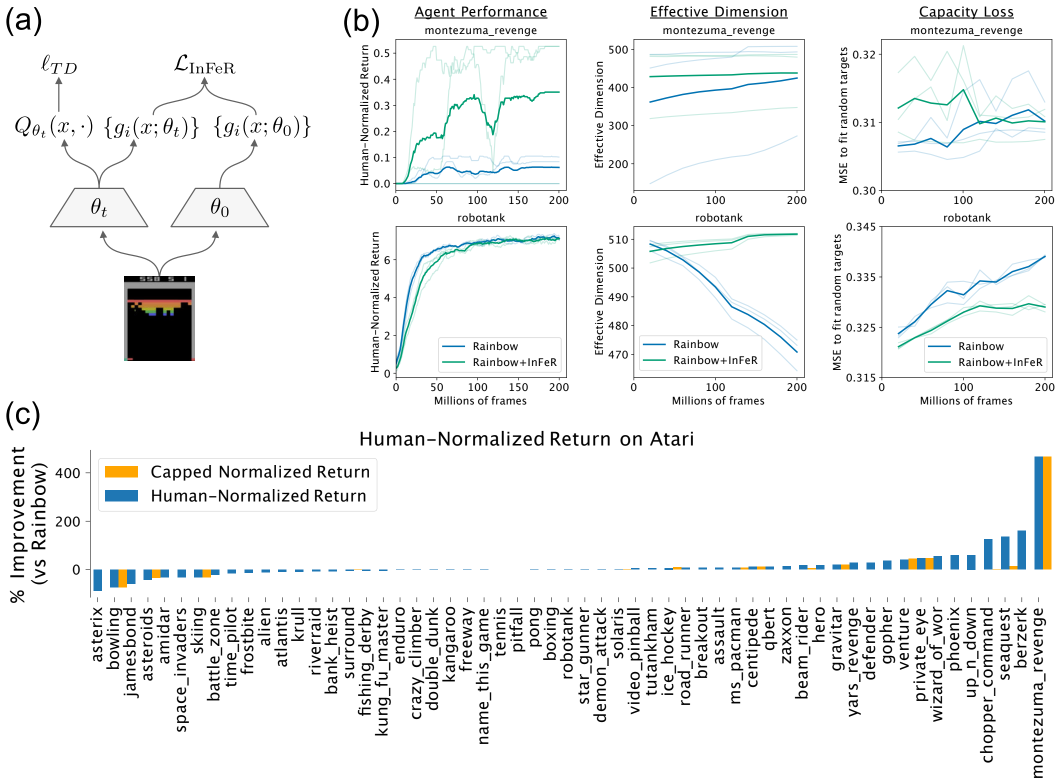

Figure 5: **(a)** Visualization of InFeR. **(b)** Analysis of capacity loss and agent performance. **(c)** Performance of InFeR relative to Rainbow on all 57 atari games.

increased this amplification effect. This results in the following form of our regularization objective, where we let $\mathcal{B}$ denote the replay buffer sampling scheme used by the agent:

$$\mathcal{L}_{\text{InFeR}}(\theta, \theta_0; \mathcal{B}, \beta) = \mathbb{E}_{x\sim\mathcal{B}}\left[\sum_{i=1}^{k}(g_i(x;\theta) - \beta g_i(x;\theta_0))^2\right] . \tag{6}$$

We evaluate the effect of incorporating this loss in both DDQN (Van Hasselt et al., 2016) and Rainbow (Hessel et al., 2018) agents, and include the relative performance improvement obtained by the InFeR agents over Rainbow on 57 games from the Atari 2600 suite in Figure 5, deferring the comparison to DDQN, where the regularizer improved performance slightly on average but only yielded significant improvements on sparse-reward games, to the appendix. We observe a net improvement over the Rainbow baseline by incorporating the InFeR objective, with significant improvements in games where agents struggle to obtain human performance. Though we only show one set of hyperparameters, we found InFeR to be relatively robust to the auxiliary scale α, target scale β, and number of heads k and so did not do extensive hyperparameter tuning. The evaluations in Figure 5 are for $k = 10$ heads with $\beta = 100$ and $\alpha = 0.1$.

We further observe in Figure 5 that in addition to improving performance, the InFeR loss reduces target-fitting error on dense-reward games and increases effective dimension in sparse-reward games, though we note some ambiguity in the target-fitting capacity results in Montezuma's Revenge, which we attribute to the fact that the sparse-reward period of training largely concludes within the first 20 million frames, well before the first agent checkpoint we consider in our effective dimension and capacity plots is saved. To fill this gap, we provide additional results in Appendix B.4 where we show that adding InFeR to the sequential MNIST prediction task reduces target-fitting error.

The striking improvement obtained in the sparse-reward Montezuma's Revenge environment begs the question of whether such results can be replicated in other RL agents. We follow the same experimental procedure as before, but now use the DDQN agent; see Figure 6. We find that adding InFeR to the DDQN objective produces a similar improvement as does adding it to Rainbow, leading the DDQN agent, which only follows an extremely naive ϵ-greedy exploration strategy and obtains zero reward at all points in training, to exceed the performance of the noisy networks approach taken by Rainbow in the last 40 million training frames. This leads to two intriguing conclusions: first, that agents which are explicitly regularized to prevent representation collapse *can* make progress in sparse reward problems without the help of good exploration strategies; and second, that this form of regularization yields significantly larger performance improvements in the presence of additional algorithm design choices that are designed to speed up learning progress.

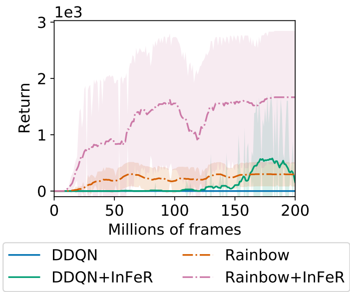

Figure 6: Additional evaluations with a Double DQN agent on Montezuma's Revenge.

4.2 Understanding How InFeR Works

Having observed that our regularizer both improves performance and mitigates capacity loss, we now take a closer look into the mechanisms by which it may enable learning progress. While InFeR improves performance *on average* across the Atari games, it does not do so uniformly: its improvements are concentrated principally on games where the baseline rainbow agent performs significantly below the human baseline. It further clearly slows down progress in a subset of environments such as Asteroids and Jamesbond. We now take a closer look at the mechanisms by which this regularizer is shaping the agent's representation in the hopes of explaining this differential effect. We consider two hypotheses.

Hypothesis 1: InFeR improves performance by preserving a random subspace of the representation that the final linear layer can use to better predict the value function. The effect of the regularizer on other aspects of the representation learning dynamics does not influence performance.

Hypothesis 2: The InFeR loss slows down the rate at which the learned features at every layer of the network can drift from their initialization in function space, improving the learning dynamics of the entire network to prevent feature collapse and over-fitting to past targets. The precise subspace spanned by the auxiliary weights is not directly useful to value function estimation.

To evaluate hypothesis 1, we concatenate the outputs of a randomly initialized network to the feature outputs of the network used to learn the Q-function, and train a linear layer on top of these joint learned and random features. If Hypothesis 1 were true, then we would expect this architecture to perform comparably to the InFeR agents, as the final linear layer has access to a randomly initialized feature subspace. If not, then we would expect the performance of the agents with access to the random features to be comparable to that of the vanilla Rainbow agents. Figure 7 shows that the latter occurs, confirming that the effect of InFeR on earlier layers is crucial to its success.

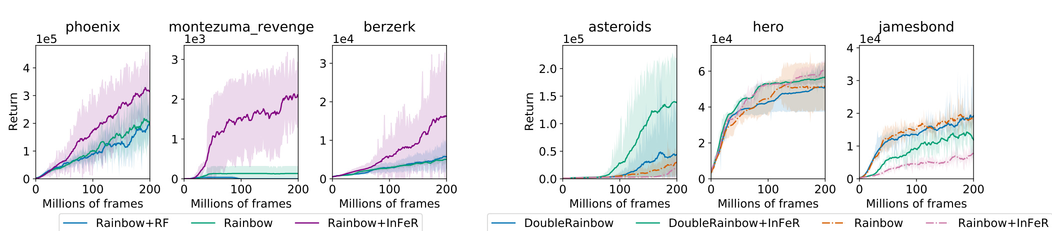

Figure 7: Left: agent performance does not improve over baseline when random features are added to the representation. Right: doubling the width of the neural network narrows the performance gap in games on which InFeR under-performed relative to Rainbow.

We now consider hypothesis 2. We note that we have constructed InFeR so as to limit the degrees of freedom with which a network can collapse its representation. This has the potential side effect of reducing the flexibility of the network to make the changes necessary to fit new value functions, which may slow down progress in environments where representation collapse doesn't occur. If this is the case, then increasing the dimension of the feature vector to which we apply InFeR should give the network more degrees of freedom to fit its targets, and so reduce or eliminate the performance gap induced by the regularization. We test this hypothesis by doubling the width of the penultimate network layer and comparing the performance of InFeR and Rainbow on games where InFeR hurt performance in the original network. We see in Figure 7 that increasing the network's size reduces, eliminates, or in some cases reverses the performance gap induced by InFeR in the smaller architecture. We therefore conclude that the principal mechanism by which InFeR affects performance is by regularizing the entire network's learning dynamics.

5 Related Work

A great deal of recent work on deep reinforcement learning has focused on developing useful auxiliary tasks to improve performance and encourage learned representations to satisfy desirable properties (Jaderberg et al., 2017; Veeriah et al., 2019; Gelada et al., 2019; Machado et al., 2018). Additional work has analyzed the geometry (Bellemare et al., 2019) and stability (Ghosh & Bellemare, 2020) of the learned features. A separate line of work considers the linear algebraic properties of agents' learned representations (Kumar et al., 2021; Gogianu et al., 2021). A theoretical framework for the analysis of agents' learning dynamics was proposed by Lyle et al. (2021), who used this to study the behaviour of RL agents in sparse-reward environments. In contrast to prior work, which treats the layers of the network which come before the features as a black box, we explicitly study the properties and learning dynamics of the whole network.

A separate line of work has studied the effect of interference between sub-tasks in both reinforcement learning (Schaul et al., 2019; Teh et al., 2017; Igl et al., 2021) and supervised learning settings (Sharkey & Sharkey, 1995; Ash & Adams, 2020; Beck et al., 2021). A great deal of work has focused on mitigating catastrophic forgetting, proposing novel training algorithms using regularization (Kirkpatrick et al., 2017; Bengio et al., 2014; Lopez-Paz & Ranzato, 2017) or distillation (Schwarz et al., 2018; Silver & Mercer, 2002; Li & Hoiem, 2017) approaches. Methods which involve re-initializing a new network have seen particular success at reducing interference between tasks in deep reinforcement learning (Igl et al., 2021; Teh et al., 2017; Rusu et al., 2016; Fedus et al., 2020). A closer relative of our approach is that of Benjamin et al. (2019), which also applies a function-space regularization approach, but which involves saving input-output pairs into a memory bank with the goal of mitigating catastrophic forgetting. InFeR differs from prior work in both goal and method: it seeks to maximize performance on *future* tasks, works without task labels, and incurs a minimal, fixed computational cost independent of the number of prediction problems seen during training.

6 Conclusions

This paper has demonstrated a fundamental challenge facing deep RL agents: loss of the capacity to distinguish states and represent new target functions over the course of training. We have shown that this phenomenon is particularly salient in sparse-reward settings, in some cases leading to complete collapse of the representation and preventing the agent from ever making learning progress. We've also proposed a regularizer to preserve capacity, yielding improved performance across a number of settings in which deep RL agents have historically struggled to match human performance. Further investigation into this method suggests that it is performing a form of function-space regularization on the neural network, and that settings where it appears the task reduces performance are actually instances of under-parameterization relative to the difficulty of the environment. Particularly notable is the effect of incorporating InFeR in the hard exploration game of Montezuma's Revenge: its success here suggests that effective representation learning can allow agents to learn good policies in sparse-reward environments even under naive exploration strategies. Our findings open up a number of exciting avenues for future work in reinforcement learning and beyond to better understand how to preserve plasticity in non-stationary prediction tasks.

References

Miguel Abreu, Luis Paulo Reis, and Nuno Lau. Learning to run faster in a humanoid robot soccer environment through reinforcement learning. In *Robot World Cup*, pp. 3–15. Springer, 2019.

Jordan Ash and Ryan P Adams. On warm-starting neural network training. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Nathan Beck, Durga Sivasubramanian, Apurva Dani, Ganesh Ramakrishnan, and Rishabh Iyer. Effective evaluation of deep active learning on image classification tasks. *arXiv*, 2021.

Marc Bellemare, Will Dabney, Robert Dadashi, Adrien Ali Taiga, Pablo Samuel Castro, Nicolas Le Roux, Dale Schuurmans, Tor Lattimore, and Clare Lyle. A geometric perspective on optimal representations for reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

Emmanuel Bengio, Joelle Pineau, and Doina Precup. Correcting momentum in temporal difference learning. In *NeurIPS Deep Reinforcement Learning Workshop*, 2020.

Yoshua Bengio, Mehdi Mirza, Ian Goodfellow, Aaron Courville, and Xia Da. An empirical investigation of catastrophic forgeting in gradient-based neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Ari Benjamin, David Rolnick, and Konrad Kording. Measuring and regularizing networks in function space. In *International Conference on Learning Representations (ICLR)*, 2019.

Will Dabney, André Barreto, Mark Rowland, Robert Dadashi, John Quan, Marc G Bellemare, and David Silver. The value-improvement path: Towards better representations for reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2021.

William Fedus, Dibya Ghosh, John D Martin, Marc G Bellemare, Yoshua Bengio, and Hugo Larochelle. On catastrophic interference in Atari 2600 games. *arXiv*, 2020.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning (ICML)*, 2019.

Dibya Ghosh and Marc G Bellemare. Representations for stable off-policy reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2020.

Florin Gogianu, Tudor Berariu, Mihaela Rosca, Claudia Clopath, Lucian Busoniu, and Razvan Pascanu. Spectral normalisation for deep reinforcement learning: an optimisation perspective. *arXiv preprint arXiv:2105.05246*, 2021.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI conference on artificial intelligence*, 2018.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations (ICLR)*, 2017.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Neural Information Processing Systems (NIPS)*, 2017.

Clare Lyle, Mark Rowland, Georg Ostrovski, and Will Dabney. On the effect of auxiliary tasks on representation dynamics. In *Artificial Intelligence and Statistics (AISTATS)*, 2021.

Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. In *International Conference on Learning Representations (ICLR)*, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Hossein Mobahi, Mehrdad Farajtabar, and Peter L Bartlett. Self-distillation amplifies regularization in Hilbert space. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017. ISSN 0036-8075.

John Quan and Georg Ostrovski. DQN Zoo: Reference implementations of DQN-based agents, 2020. URL `http://github.com/deepmind/dqn_zoo`.

Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in reinforcement learning. *arXiv*, 2020.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *International Conference on Learning Representations (ICLR)*, 2016.

Tom Schaul, Diana Borsa, Joseph Modayil, and Razvan Pascanu. Ray interference: a source of plateaus in deep reinforcement learning. *arXiv*, 2019.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning (ICML)*, 2018.

Noel E Sharkey and Amanda JC Sharkey. An analysis of catastrophic interference. *Connection Science*, 1995.

Daniel L Silver and Robert E Mercer. The task rehearsal method of life-long learning: Overcoming impoverished data. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pp. 90–101, 2002.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Yee Whye Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Neural Information Processing Systems (NIPS)*, 2017.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *AAAI Conference on Artificial Intelligence*, 2016.

Vivek Veeriah, Matteo Hessel, Zhongwen Xu, Richard Lewis, Janarthanan Rajendran, Junhyuk Oh, Hado van Hasselt, David Silver, and Satinder Singh. Discovery of useful questions as auxiliary tasks. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

A Estimator Consistency

We here show that our estimator of the agent's effective dimension is consistent. First recall

$$\left(\frac{1}{\sqrt{n}}\Phi_n\right)^\top \left(\frac{1}{\sqrt{n}}\Phi_n\right) = \frac{1}{n}\sum_{i=1}^{n} \phi(x_i)\phi(x_i)^\top . \tag{7}$$

The following property of the expected value holds

$$\mathbb{E}_{x\sim P}[\phi(x)\phi(x)^\top] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\phi(x_i)\phi(x_i)^\top\right] . \tag{8}$$

It is then straightforward to apply the strong law of large numbers. To be explicit, we consider an element of $M = \mathbb{E}[\phi\phi^\top]$, M_{ij}.

$$\mathbb{E}[(\phi(x)\phi(x)^\top)_{ij}] = M_{ij} = \mathbb{E}[\phi_i(x)\phi_j(x)] \implies \sum_{k=1}^{n}\frac{1}{n}\phi_i(x_k)\phi_j(x_k) \overset{a.s.}{\to} M_{ij} . \tag{9}$$

Since we have convergence for any M_{ij}, we get convergence of the resulting matrix to M. Because the singular values of Φ are the eigenvalues of M and the eigenvalues are continuous functions of that matrix, the eigenvalues of M_n converge to those of M in probability. Then for almost all values of ϵ, the threshold estimator $N(\lambda_1, \dots, \lambda_k; \epsilon) = |\{\lambda_i > \epsilon\}|$ will converge to $N(\text{spec}(M); \epsilon)$ for almost all values ϵ. Specifically, the estimator will be convergent for all values of ϵ which are not eigenvalues of M itself.

B Additional Evaluations

We now present full evaluations of many of the quantities described in the paper. We use the same training procedure for all of the figures in this section, loading agent parameters from checkpoints to compute the quantities shown.

- **Agent:** We train a Rainbow agent (Hessel et al., 2018) with the same architecture and hyper-parameters as are described in the open-source implementation made available by Quan & Ostrovski (2020). We additionally add InFeR, as described in Section 4, with 10 heads, gradient weight 0.1 and scale 100.
- **Training:** We follow the training procedure found in the Rainbow implementation mentioned above. We train for 200 million frames, with 500K evaluation frames interspersed every 1M training frames. We save the agent parameters and replay buffer every 10M frames to estimate feature dimension and target-fitting capacity.

B.1 Effective Dimension

Here we expand upon the illustrative examples provided in Figure 3. We show that the decline in dimension shown across the different agents in the selected games also occurs more generally in Rainbow agents across most environments in the Atari benchmark. We also show that in most cases adding InFeR mitigates this phenomenon. Our observations here do not show a uniform decrease in effective dimension or a uniformly beneficial effect of InFeR. The waters become particularly muddied in settings where neither the Rainbow nor Rainbow+InFeR agent consistently make learning progress such as in tennis, solaris, and private eye. It is outside the scope of this work to identify precisely why the agents do not make learning progress in these settings, but it does not appear to be due to the type of representation collapse that can be effectively prevented by InFeR.

Procedure. We compute the effective dimension by sampling $n = 5e^4$ transitions from the replay buffer and take the set of origin states as the input set. We then compute a $n \times d$ matrix whose row i is given by the output of the penultimate layer of the neural network given input S_i. We then take the singular value decomposition of this matrix and count the number of singular values greater than 0.01 to get an estimate of the dimension of the network's representation layer.

In most games, we see a decline in effective dimension. Strikingly, this decline in dimension holds even in the online RL setting where the agent's improving policy presumably leads it to observe a more diverse set of states over time, which under a fixed representation would tend to increase the effective rank of the feature matrix. This indicates that even in the face of increasing state diversity, agents' representations face strong pressure towards degeneracy.

B.2 Target-fitting Capacity in Atari

Analogously, we also evaluate the target-fitting capacity of Rainbow and Rainbow+InFeR agents across the entire Atari suite. We again observe that in general the target-fitting error of the network-optimizer combination tends to increase slightly over the course of training. In most cases, we see a modest increase in target-fitting error which is somewhat reduced by adding InFeR, although both the overall trend over the course of training and the effect of InFeR vary between the different games.

Procedure. We measure target-fitting capacity by using the outputs of a randomly initialized network of the same architecture as the Rainbow agent as regression targets. At each saved checkpoint, we load the agent's parameters, optimizer state, and the replay buffer from either the same checkpoint as the parameters and optimizer state or from the *first* checkpoint. We then randomly sample $n = 2000$ transitions and perform ℓ_2 regression on the outputs of the randomly initialized network, using the checkpoint parameters and optimizer state as an initialization point. We train for 200 epochs using the same optimizer as was used in RL training, and compute the mean squared error from the targets at the end of training as the target-fitting error.

Intriguingly, the set of games which see decreases in effective dimension and the set of games which see increases in target-fitting error overlap but are not identical. There are a number of possible explanations for why this may be the case, which we detail below.

- **Input state similarity:** all inputs in the game of Pong are extremely similar, and therefore yield similar outputs from the randomly initialized network which pose a simpler prediction problem than those of other games where inputs are more visually diverse.
- **Training duration:** some games require longer to master than others, and so in some easy environments which plateau at the optimal policy quickly the effective training time will be much shorter than the wall-clock time, as after it has learned the optimal value function the agent receives near-zero gradients.
- **Target magnitude:** some environments have smaller target functions than others, and while the loss function appeared to plateau well before we halted training on random targets, it's possible that some environments appeared to have inflated capacity loss because it is harder to move parameters which produce large-magnitude outputs towards the random initialization even if they're equally capable of adapting to smaller changes in the target value.

B.3 Performance

We provide full training curves for both Rainbow and Rainbow+InFeR on all games in Figures 10 & 11 (capped human-normalized performance), and 12 & 13 (raw evaluation score). We also provide evaluation performance curves for DDQN and DDQN+InFeR agents in Figure 14.

B.4 Details: Target-fitting Capacity in Non-stationary MNIST

In addition to our evaluations in the Atari domain, we also consider a variant of the MNIST dataset in which the labels change over the course of training.

- **Inputs and Labels:** We use 2000 randomly sampled digits from the MNIST dataset and assign either binary or random Gaussian targets.
- **Distribution Shift:** We divide training into 10 stages of 50 epochs each. In the Gaussian setting, the labels are resampled every stage. In the binary setting, the labels at stage i correspond to the

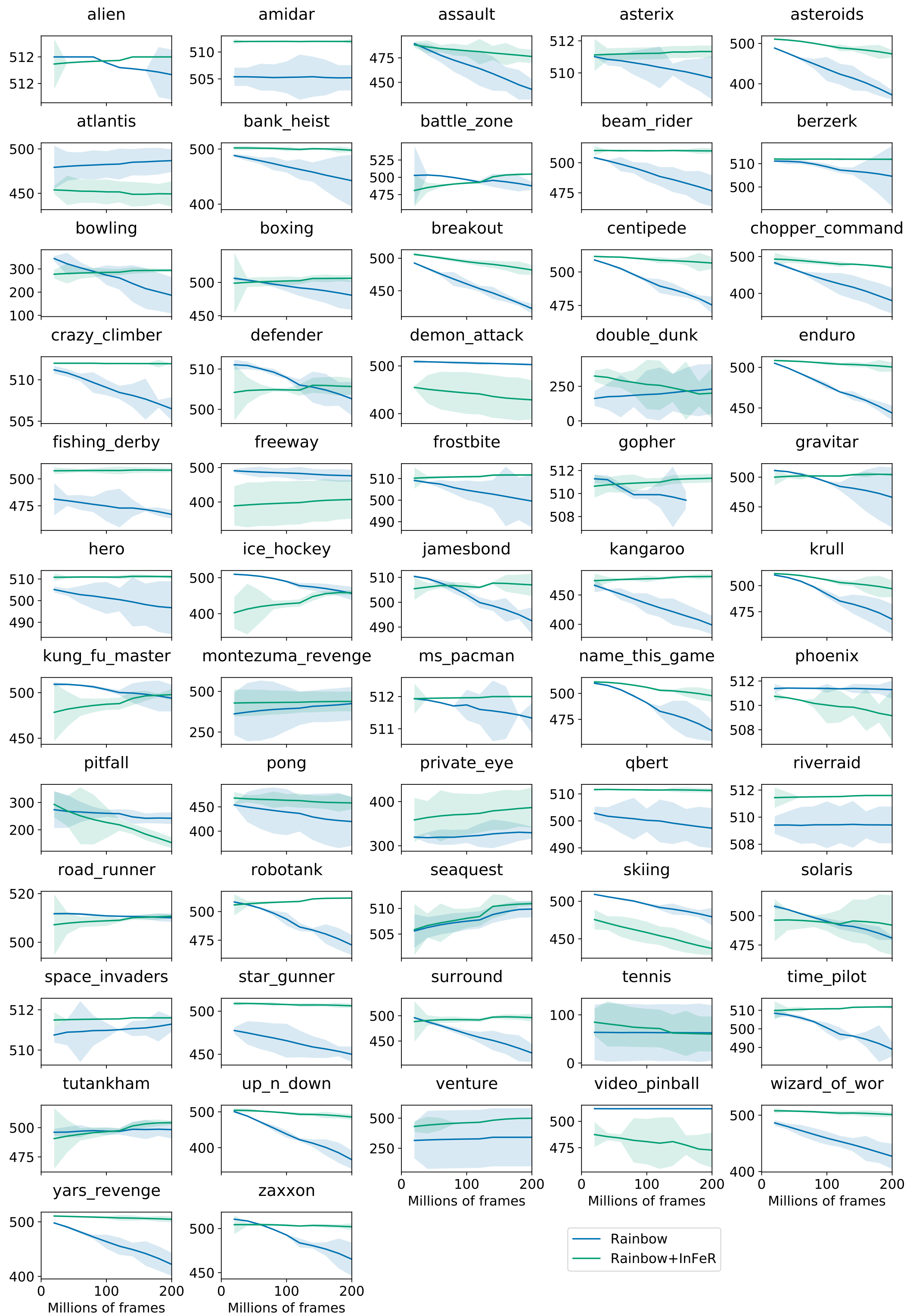

Figure 8: Effective dimension of agent representations over the course of training on all 57 games in the Atari benchmark. We compare Rainbow against Rainbow+InFeR. Rainbow+InFeR does not uniformly prevent decreases in effective dimension across all games, but on average it has a beneficial effect on preserving representation dimension.

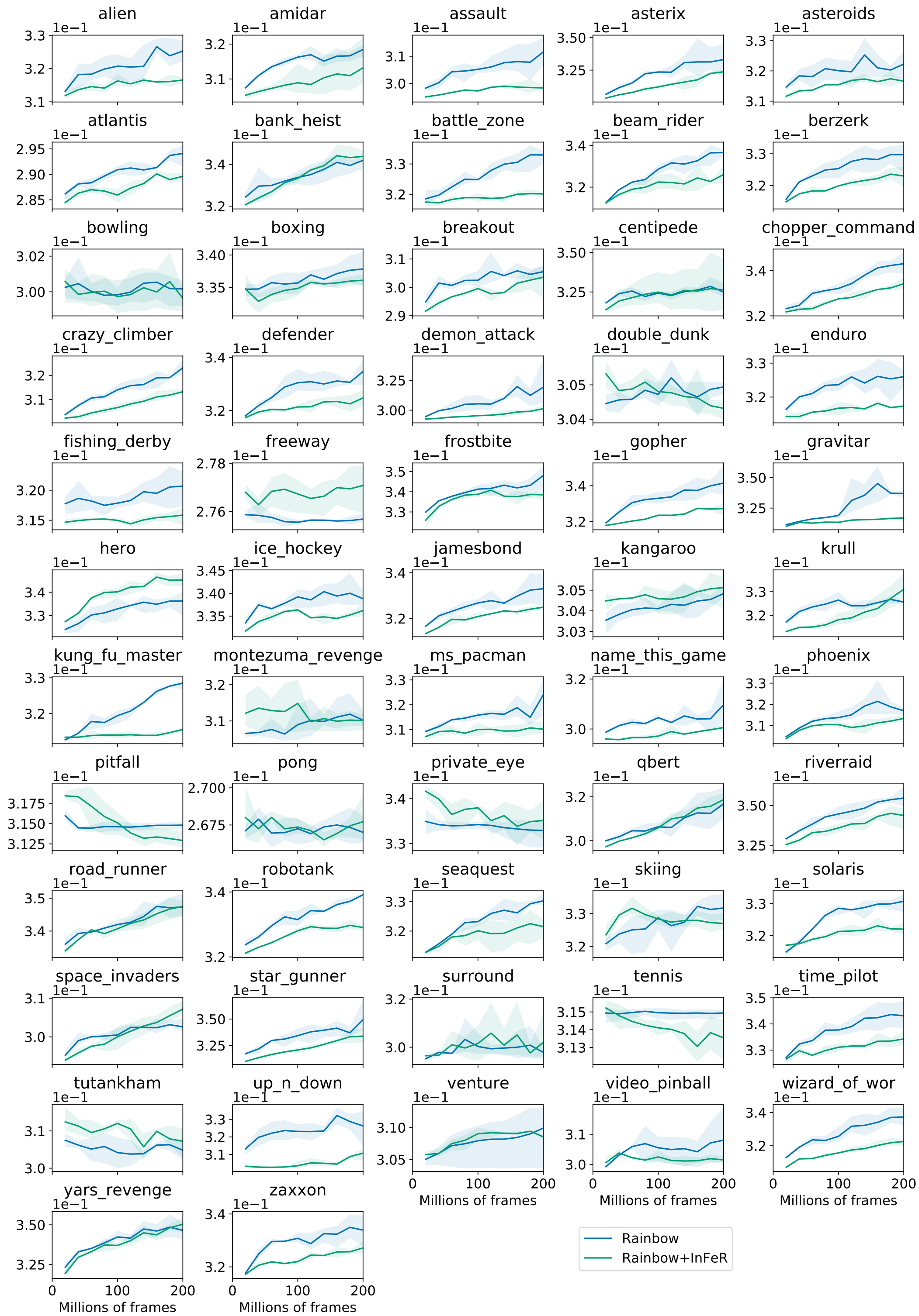

Figure 9: Target-fitting capacity (MSE on random targets) measured over the course of training on all games in the Atari benchmark. In most (43 out of 57) environments, target-fitting capacity declines over the course of training in both standard Rainbow and Rainbow+InFeR, but in these settings the Rainbow+InFeR agent achieves lower error in almost all environments.

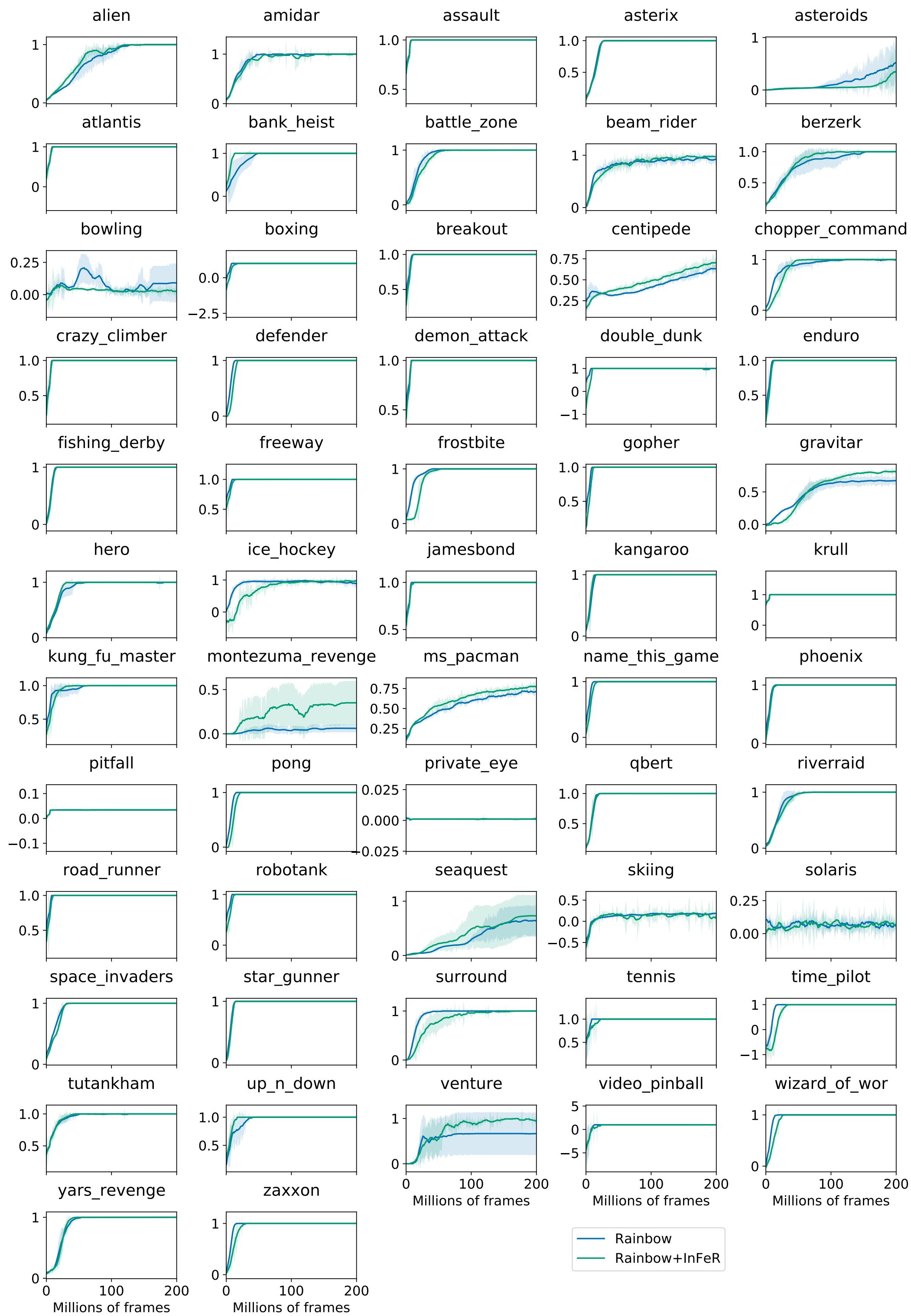

Figure 10: Full evaluation of capped human-normalized performance on Atari benchmarks for the default Rainbow architecture.

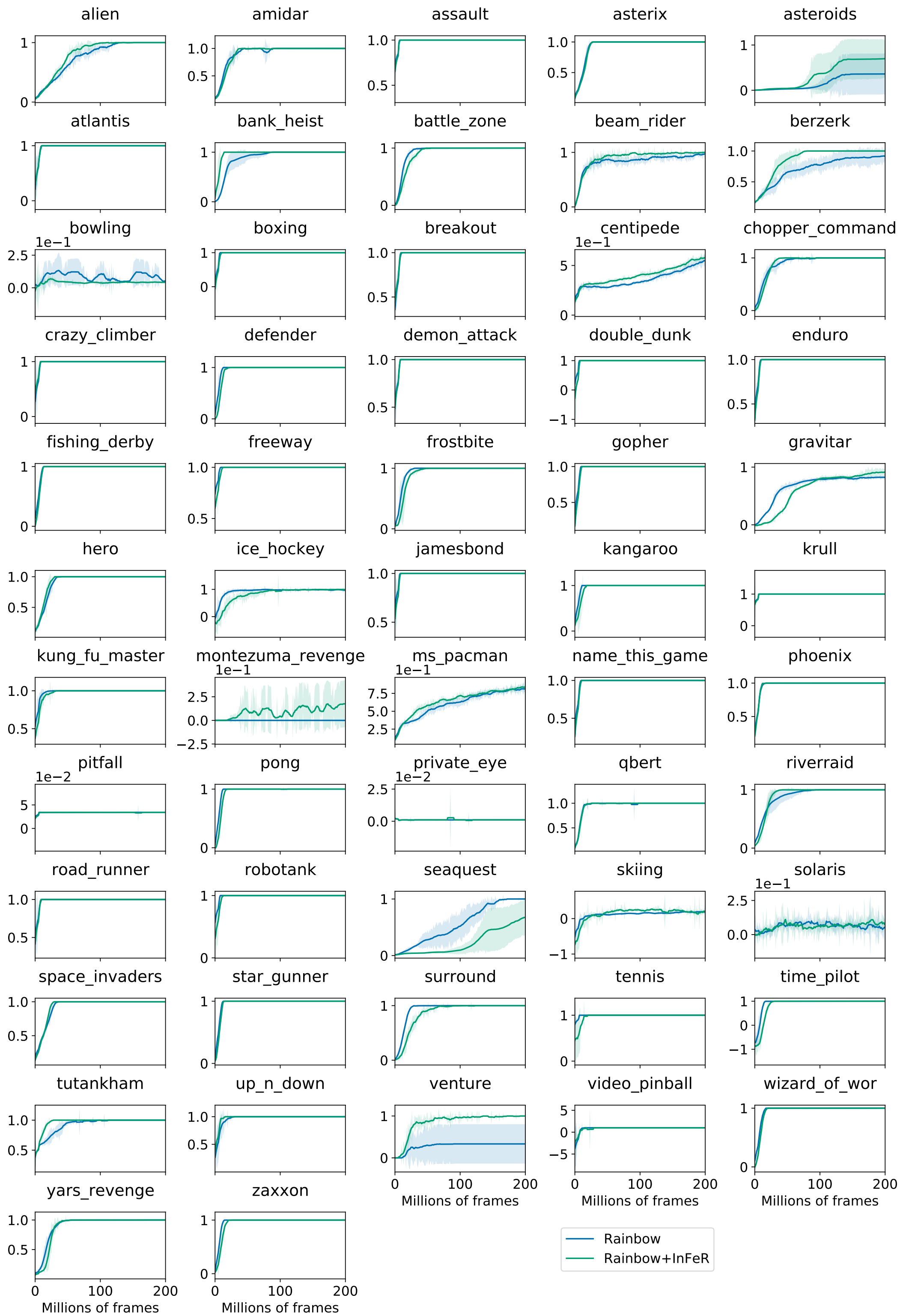

Figure 11: Full evaluation of capped human-normalized performance on Atari benchmarks in the double-width Rainbow architecture.

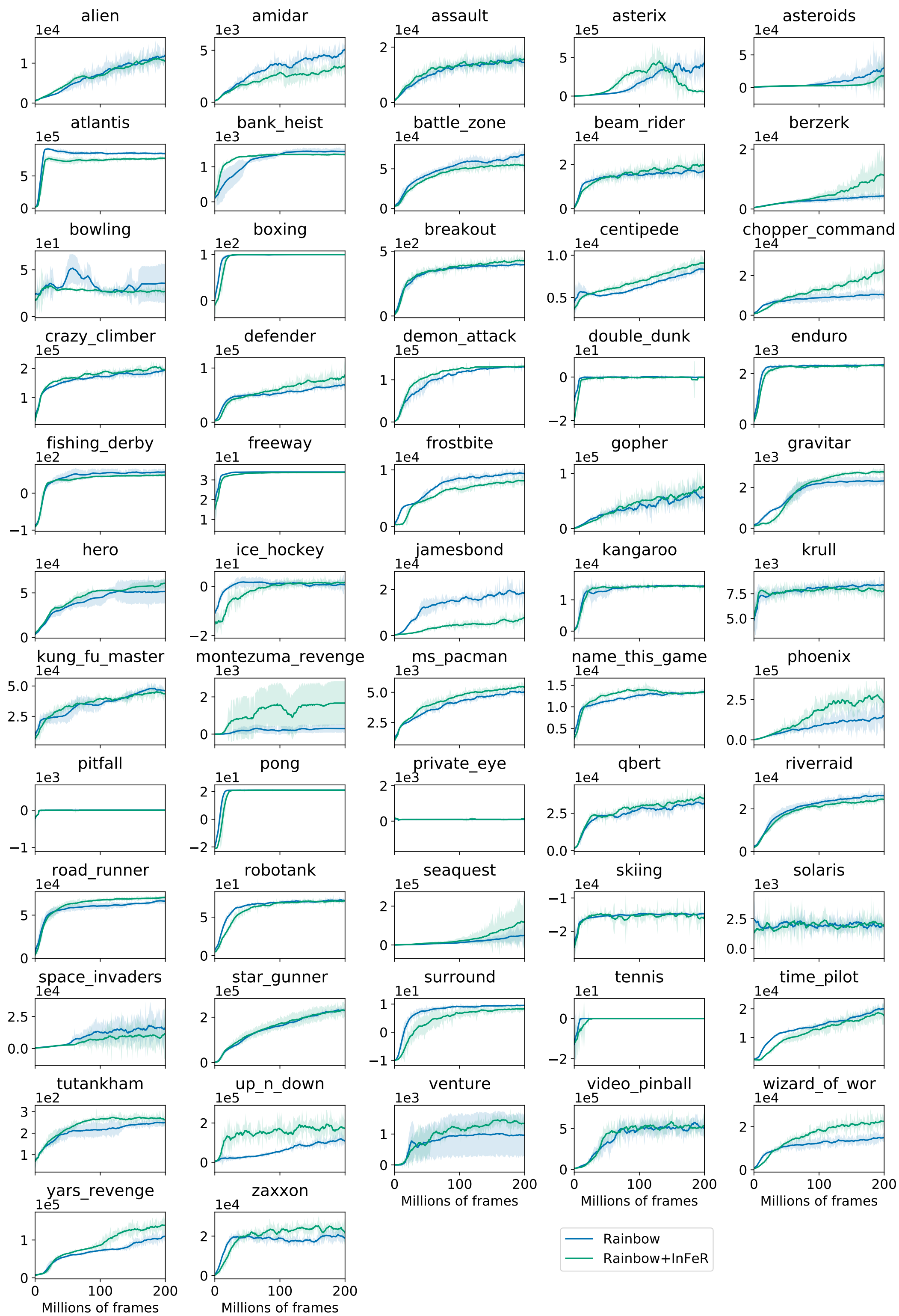

Figure 12: Full evaluation of raw scores on Atari benchmarks for the default Rainbow architecture.

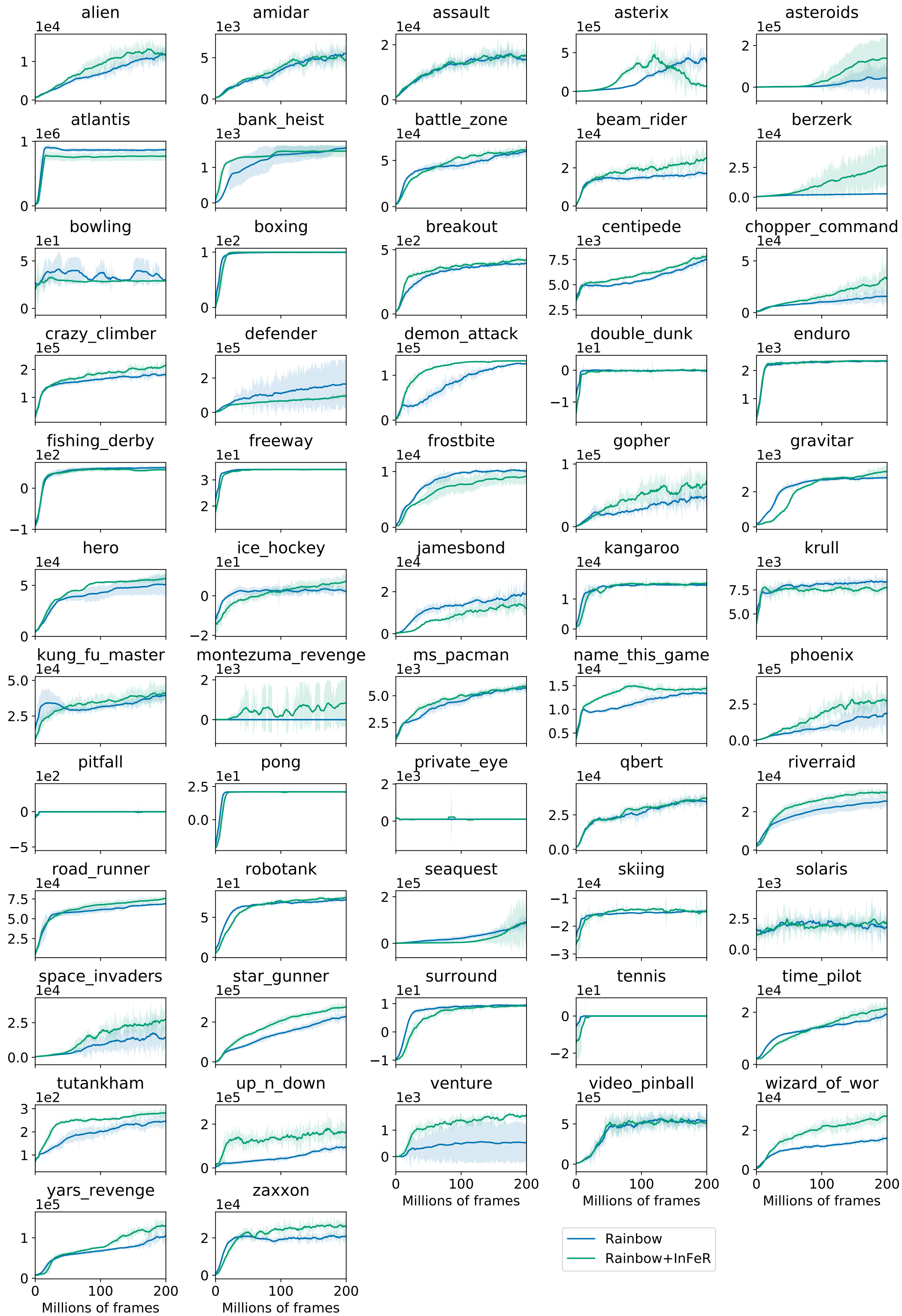

Figure 13: Full evaluation of raw scores on Atari benchmarks for the double-width Rainbow architecture.

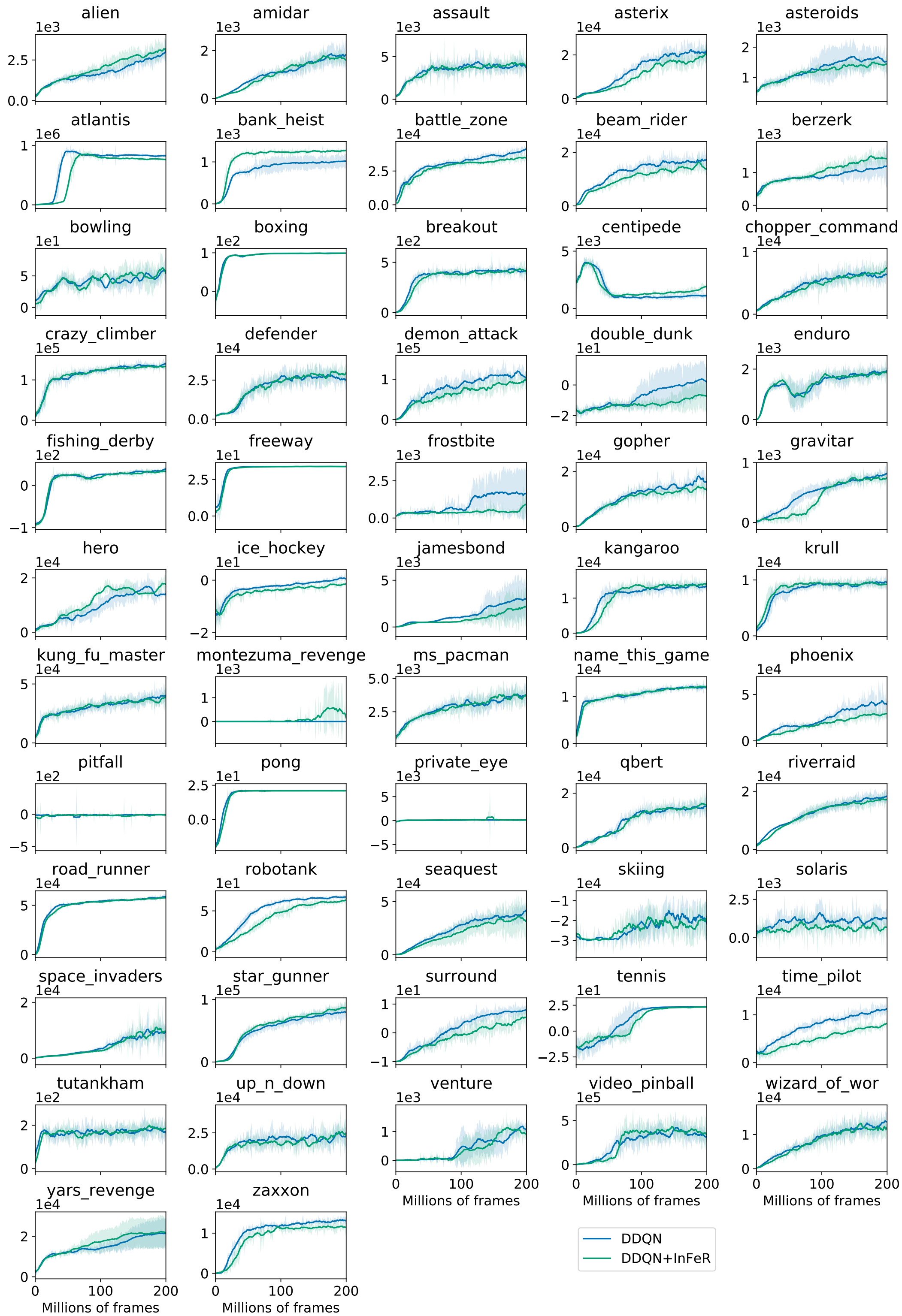

Figure 14: Evaluations of the effect of InFeR on performance of a Double DQN agent. Overall we do not see as pronounced an improvement as in Rainbow, but note that the average human-normalized score over the entire benchmark is nonetheless slightly higher for the InFeR agent, and that the performance improvement obtained by InFeR in Montezuma's Revenge is still significant in this agent.

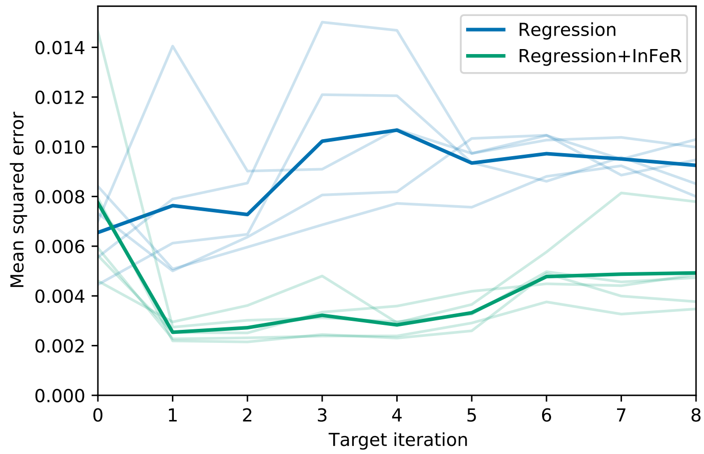

Figure 15: Effect of adding InFeR to the regression objective in a random reward prediction problem on the non-stationary MNIST environment. We see that the InFeR objective produces networks that can consistently outperform those trained with a standard regression objective, and even appears to enable forward-transfer early in training.

binary indicator $\mathbb{1}[y > i]$. This mimics the phenomenon in RL where states' values may increase over time as the agent's policy improves.

- **Architecture:** we use two different MLP architectures: a 'shallow' single hidden layer network with width 128, along with a 'deep' MLP with hidden unit widths 64 and 32, along with two activation units: ReLU and leaky-ReLU.

Our results from this experiment indicate that underparameterized networks and those that use ReLU activations are more prone to capacity loss than networks with abundant capacity and which do not use ReLU activations.

B.5 Effect of InFeR on target-fitting capacity in MNIST

In addition to our study of the Atari suite, we also study the effect of InFeR on the non-stationary MNIST reward prediction task with a fully-connected architecture; see Figure 15. We find that it significantly mitigates the decline in target-fitting capacity demonstrated in Figure 2.