A THEORETICAL RESULTS

A.1 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 almost surely. Then for almost all values of ϵ, the threshold estimator $N(\lambda_1, \ldots, \lambda_k; \epsilon) = |\{\lambda_i > \epsilon\}|$ will converge to $N(\text{spec}(M); \epsilon)$. Specifically, the estimator will be convergent for all values of ϵ which are not eigenvalues of M itself.

A.2 FEATURE DYNAMICS

We apply similar analysis to that of Lyle et al. (2021) to better understand the effect of sparse-reward environments on representation collapse. To do so, we consider the setting where Φ_t are features and w_t a linear function approximator which jointly parameterize a value function $V_t = \langle \Phi_t(x), w_t \rangle$. We will be interested in studying a continuous-time approximation to TD learning, where the discrete-time expected updates

$$\Phi_t \leftarrow \Phi_t + \alpha\nabla_\Phi V_t[(\gamma P^\pi - I)V_t + R^\pi] \tag{10}$$

$$w_t \leftarrow w_t + \beta\nabla_w V_t(\gamma P^\pi - I)V_t + R^\pi] \tag{11}$$

are translated into a continuous-time flow, described by the following equations.

$$\partial_t\Phi_t = \alpha(\gamma P^\pi - I)\Phi_t(w_t w_t^\top) + R^\pi w_t^\top \tag{12}$$

$$\partial_t w_t = \beta\Phi_t^\top[(\gamma P^\pi - I)\Phi_t w_t + R^\pi] , \tag{13}$$

where $P^\pi \in \mathbb{R}^{\mathcal{X}\times\mathcal{X}}$ is the matrix of state-transition probabilities under π, and $R^\pi \in \mathbb{R}^{\mathcal{X}}$ is the vector of expected rewards.

One of the key take-aways of prior works is that under certain assumptions, a tabular value function following continuous-time TD dynamics will converge to its limiting value V^π along the principal components of the environment's transition matrix. In the function-approximation case described above, the dynamics of the features Φ_t are somewhat more complex. However, it turns out that under certain training regimes, we can obtain similar convergence results for the features. We therefore turn our attention to ensemble prediction, where M linear prediction 'heads', each using a separate weight vector w_t^m $(m = 1, \ldots, M)$ are all trained to regress on the TD targets using the shared feature representation of the state as input, resulting in the following dynamics.

$$\partial_t\Phi_t^M = \alpha\sum_{m=1}^{M}(R^\pi + \gamma P^\pi\Phi_t^M w_t^m - \Phi_t^M w_t^m)(w_t^m)^\top , \tag{14}$$

$$\partial_t w_t^m = \beta(\Phi_t^M)^\top(R^\pi + \gamma P^\pi\Phi_t^M w_t^m - \Phi_t w_t^m) . \tag{15}$$

We now restate the result of Lyle et al. (2021) regarding the behaviour of the representation in the limit of many ensemble heads.

Theorem 1 (Lyle et al., 2021)**.** *For $M \in \mathbb{N}$, let $(\Phi_t^M)_{t\geq 0}$ be the solution to Equation 14, with each w_t^m for $m = 1, \ldots, M$ initialised independently from $N(0, \sigma_M^2)$, and fixed throughout training ($\beta = 0$). We consider two settings: first, where the learning rate α is scaled as $\frac{1}{M}$ and $\sigma_M^2 = 1$ for all M, and second where $\sigma_M^2 = \frac{1}{M}$ and the learning rate α is equal to 1. These two settings yield the following dynamics, respectively:*

$$\lim_{M\to\infty} \partial_t \Phi_t^M \overset{P}{=} -(I - \gamma P^\pi)\Phi_t^M \quad \text{, and} \tag{16}$$

$$\lim_{M\to\infty} \partial_t \Phi_t^M \overset{D}{=} -(I - \gamma P^\pi)\Phi_t^M + R^\pi \epsilon^\top \text{ , } \epsilon \sim \mathcal{N}(0, I)\,. \tag{17}$$

The corresponding limiting trajectories for a fixed initialisation $\Phi_0 \in \mathbb{R}^{\mathcal{X}\times d}$, are therefore given respectively by

$$\lim_{M\to\infty} \Phi_t^M \overset{P}{=} \exp(-t(I - \gamma P^\pi))\Phi_0 \quad \text{, and} \tag{18}$$

$$\lim_{M\to\infty} \Phi_t^M \overset{D}{=} \exp(-t(I - \gamma P^\pi))(\Phi_0 - (I - \gamma P^\pi)^{-1} R^\pi \varepsilon^\top) + (I - \gamma P^\pi)^{-1} R^\pi \varepsilon^\top \text{ , } \epsilon \sim \mathcal{N}(0, I)\,. \tag{19}$$

One important corollary of this result occurs in sparse-reward environments under sub-optimal policies, where $R^\pi = \mathbf{0}$. In this case, we see that the representation converges precisely to the zero vector.

Corollary 1. *Let Φ_t^M, $(w)_{i=1}^M$ be defined as in Theorem 1. Then if $R^\pi = 0$, the feature representation converges to the zero vector for every state, independent of whether the learning rate α is scaled as $\frac{1}{M}$ or the linear weight initialization variance scales as $\frac{1}{M}$. In particular:*

$$\lim_{t\to\infty} \lim_{M\to\infty} \Phi_t^M \overset{P}{=} \mathbf{0}\,. \tag{20}$$

As a result, we have that the effective dimension of Φ will also tend to zero

$$\forall \epsilon > 0 \quad \lim_{t\to\infty} \lim_{M\to\infty} |\{\sigma \in SVD(\Phi_t^M) | \sigma > \epsilon\}| \overset{P}{=} 0\,. \tag{21}$$

Proof. The proof of this result follows from a straightforward application of Theorem 1, setting $R^\pi = 0$ and letting $t \to \infty$. We can obtain an analogous result for the srank of Φ_t^M when P^π is diagonalizable by noting that for any eigenvector v_i of P^π, the value of $v_i^\top \Phi_t^M v_i$ evolves as $c\exp(-t\lambda_i)$ for some constant c that depends on Φ_0^M. In this case, we obtain a limiting value of 1 for the srank so long as P^π corresponds to an ergodic Markov chain. □

The setting of this result is distinct from that of deep neural network representation dynamics, as neural networks use discrete optimization steps, finite learning rates, and typically do not leverage linear ensembles. However, we emphasize two crucial observations that suggest the intuition developed in this setting may be relevant: first, in sparse reward environments the representation will be pushed to zero along dimensions spanned by the linear weights used to compute outputs. Once sufficiently many independent weight vectors are being used to make predictions, this effectively forces *every* dimension of the representation to fit the zero vector output. We would therefore expect representation collapse to be particularly pronounced in the QR-DQN agents trained on sparse-reward environments, as in this setting we obtain many independently initialized heads all identically trying to fit the zero target.

Second, in the presence of ReLU activations and stochastic optimization, the trajectories followed by the learned features in deep neural networks run the risk of getting 'trapped' in negative values. If these features would normally tend to small values close to zero (as we would expect in agents following similar dynamics to those obtained in Theorem 1), this increases the risk of unit saturation, where the representation may get trapped in bad local minima. This appears to be what happens in the QR-DQN agents trained on sparse-reward environments such as Montezuma's Revenge.

Figure 7: Effective dimension and target-fitting capacity of a single training seed early in the training period. We see that early in training agents quickly diverge from the target-fitting capacity of random initializations. In dense-reward environments, these agents tend to pick obtain higher effective dimension, but in sparse-reward environments the effective dimension decreases rapidly. This figure will be updated in later revisions to include a greater range of agents and seeds, and to include evaluations from zero.

B Additional Empirical Analysis

B.1 Early Training Curves and Capacity Loss

We include here additional empirical evaluations to provide insight into how target-fitting capacity and effective dimension evolve over the course of training. For completeness, we also show that the same trends present in our estimator of effective dimension also hold for srank (Kumar et al., 2021). Based on our insights from Section A.2, our current results focus on the QR-DQN agent as this is the agent in which we expect capacity loss to be most pronounced and therefore easiest to measure. We focus on agent checkpoints from the first 10M frames of training, and use as baseline randomly initialized parameters evaluated on the same target-fitting task. We collect agent checkpoints every 1M frames, and visualize the evolution of the different measures of capacity evaluated in the main paper over the course of this early training period. We make two principal observations.

- **The feature rank (both srank and effective dimension) evolves rapidly in the early training period.** The figures from the main paper show that after the first 10M frames, feature rank seems to stabilize and changes more slowly with time. We do, however, see differences in the value attained by different algorithms in the first 10M frames, which implies that the early training period is crucial for the evolution of feature rank. The top two rows of Figure 7 show this: under both estimators, the feature rank increases over the course of training in dense-reward games, and decreases rapidly in the sparse-reward environment.
- **Target-fitting capacity also exhibits modest declines early in training.** This seems to be more pronounced in the dense-reward agents, consistent with the hypothesis that it is fitting a sequence of target functions that reduces capacity. We also note, however, that the structure of the functions that the dense- and sparse-reward agents are optimized to respectively fit differ

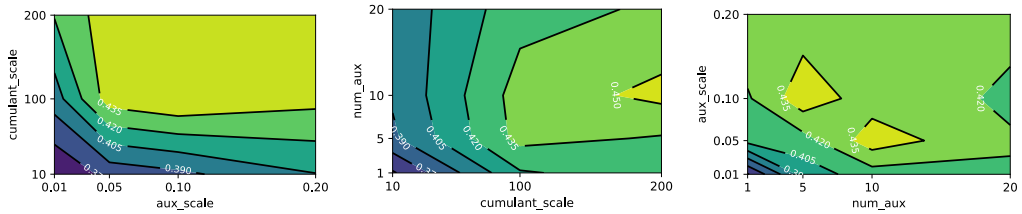

Figure 8: Hyperparameter sweeps for the DDQN+InFeR agent. Each contour plot shows average capped human-normalized score at the end of training marginalized over all hyperparameters not shown on its axes.

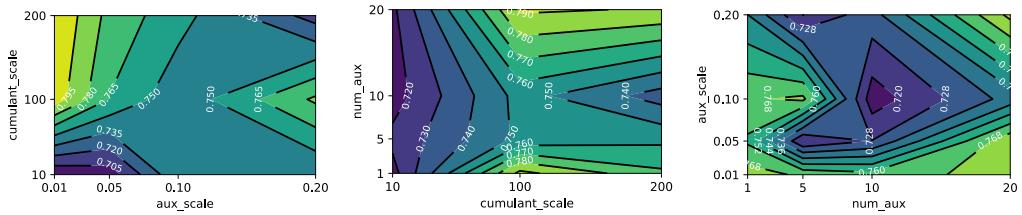

Figure 9: Hyperparameter sweeps for the Rainbow+InFeR agent. Each contour plot shows average capped human-normalized score at the end of training marginalized over all hyperparameters not shown on its axes.

significantly in their similarity to the task of fitting a randomly initialized network. As a result, the greater apparent decline in target-fitting capacity in the dense agents could also be an artifact of the greater distribution shift in the targets these agents are being asked to fit. We think this is an intriguing direction for future work.

B.2 Hyperparameter sensitivity of InFeR in deep reinforcement learning agents

We report results of hyperparameter sweeps over the salient hyperparameters relating to InFeR, so as to assess the robustness of the method. For both the DDQN and Rainbow agents augmented with InFeR, we sweep over the number of auxiliary predictions (1, 5, 10, 20), the cumulant scale used in the predictions (10, 100, 200), and the scale of the auxiliary loss (0.01, 0.05, 0.1, 0.2). We consider the capped human-normalized return across four games (Montezuma's Revenge, Hero, James Bond, and MsPacman), and run each hyperparameter configuration with 3 seeds. Results are shown in Figure 8 for the DDQN agent; we compare performance as each pair of hyperparameters varies (averaging across the other hyperparameter, games, and seeds, and the last five evaluation runs of each agent). Corresponding results for Rainbow are given in Figure 9.

C 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 hyperparameters 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.

C.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.

C.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.

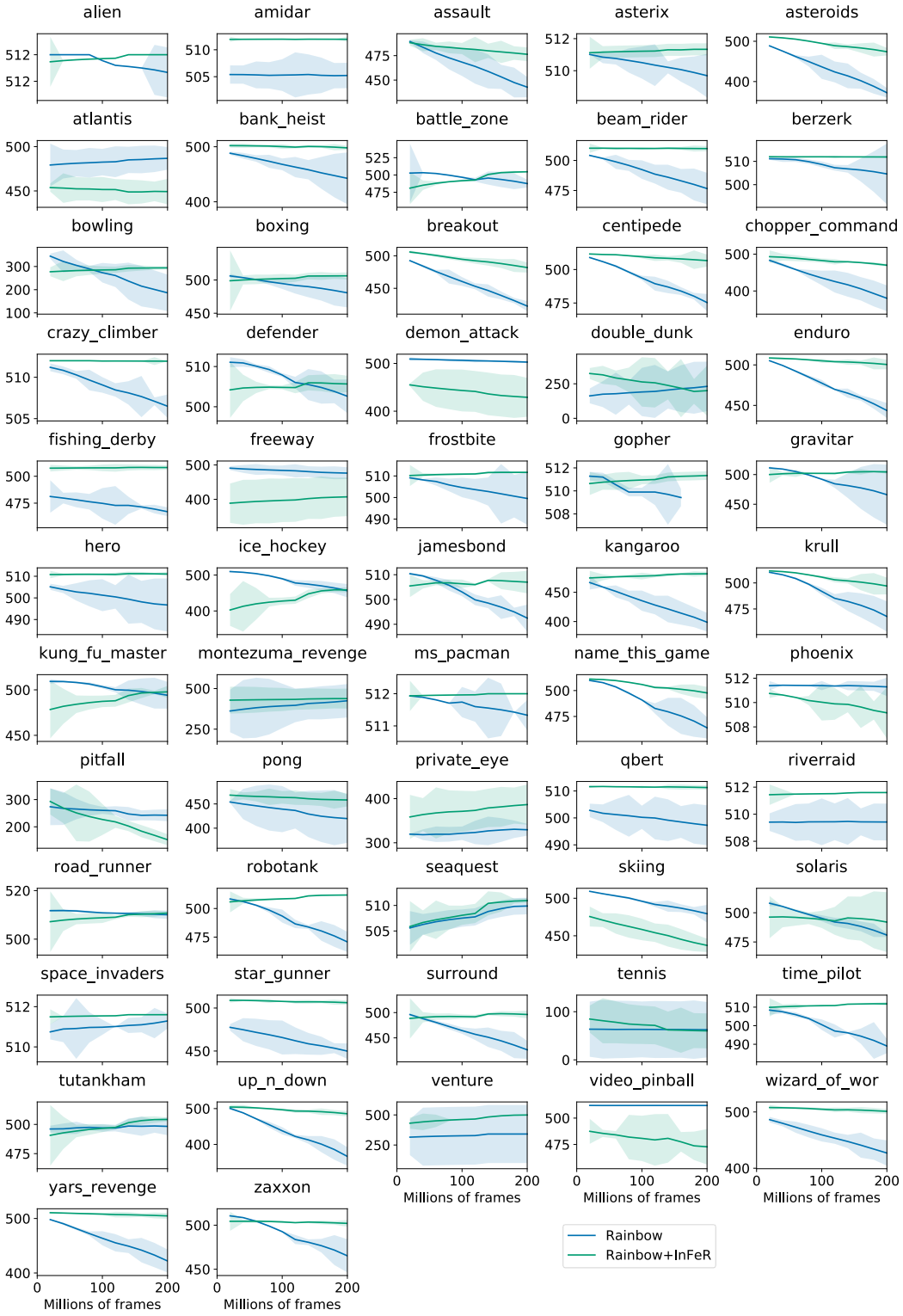

Figure 10: Effective dimension of agent representations over the course of training on all 57 games in the Atari benchmark. We compare Rainbow against Rainbow+InFeR. Rainbow+InFeR does not uniformly prevent decreases in effective dimension across all games, but on average it has a beneficial effect on preserving representation dimension.

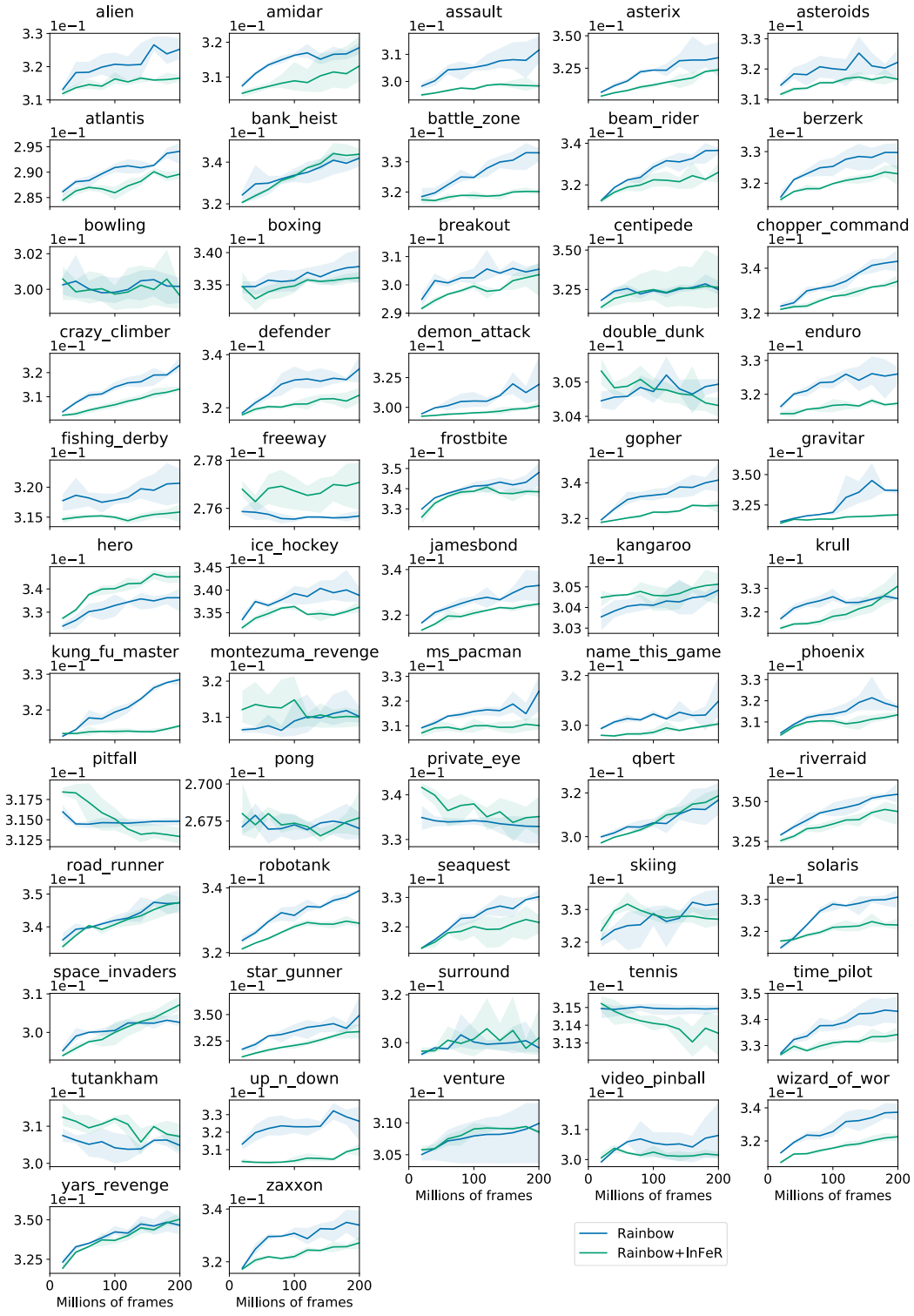

Figure 11: Target-fitting capacity (MSE on random targets) measured over the course of training on all games in the Atari benchmark. In most (43 out of 57) environments, target-fitting capacity declines over the course of training in both standard Rainbow and Rainbow+InFeR, but in these settings the Rainbow+InFeR agent achieves lower error in almost all environments.

C.3 PERFORMANCE

We provide full training curves for both Rainbow and Rainbow+InFeR on all games in Figures 12 & 13 (capped human-normalized performance), and 14 & 15 (raw evaluation score). We also provide evaluation performance curves for DDQN and DDQN+InFeR agents in Figure 16.

C.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 binary indicator $\mathbb{1}[y < i]$. In particular, this means the dataset will be of the form of input label pairs (x, y), of the form $(x, 0)$ if x corresponds to a digit with value less than i and $(x, 1)$ if x is an image of a digit with value greater than y. Over the course of this noonstationary prediction task, the number of non-zero labels strictly increases. 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.

C.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 17. We find that it significantly mitigates the decline in target-fitting capacity demonstrated in Figure 2.

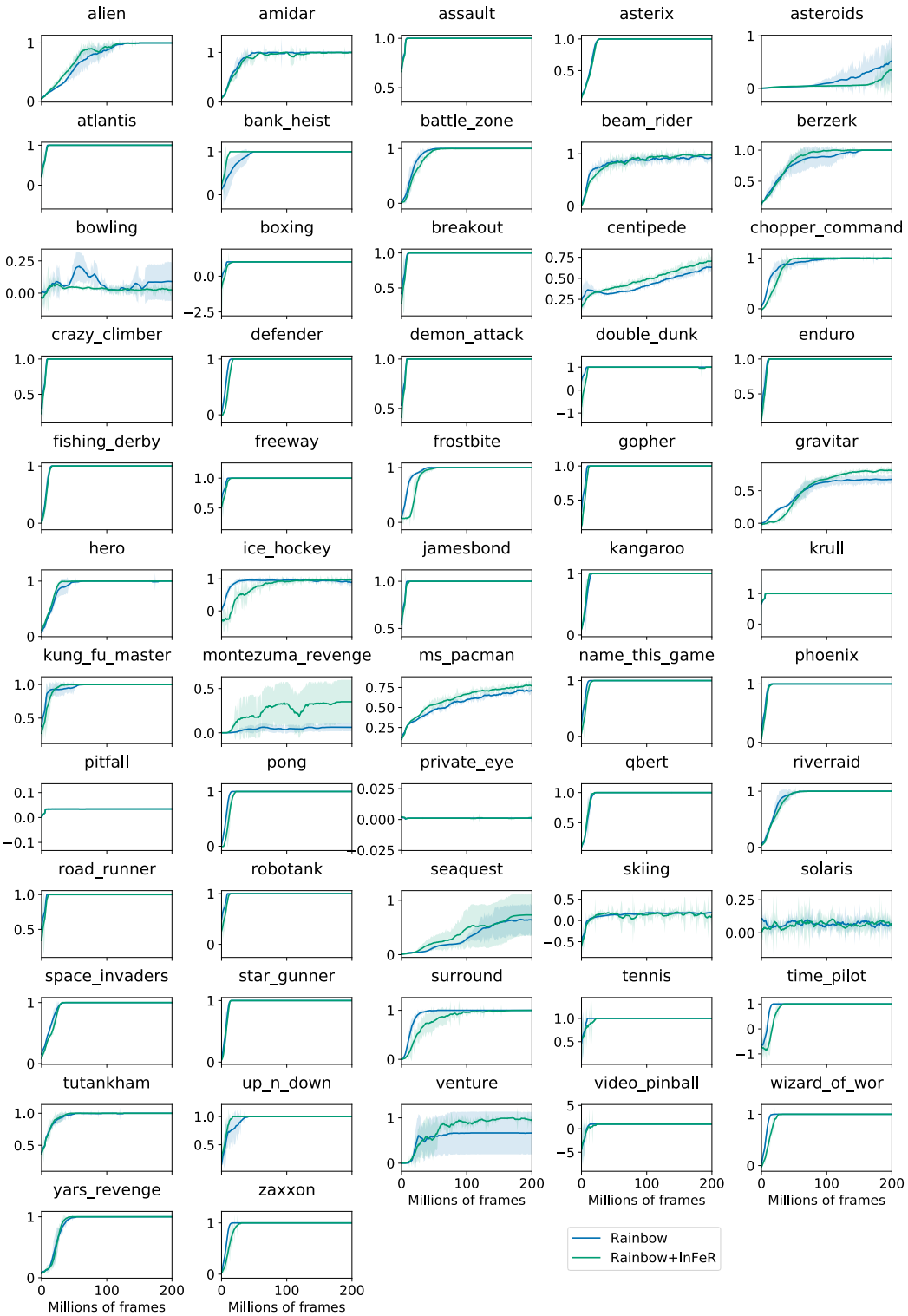

Figure 12: Full evaluation of capped human-normalized performance on Atari benchmarks for the default Rainbow architecture.

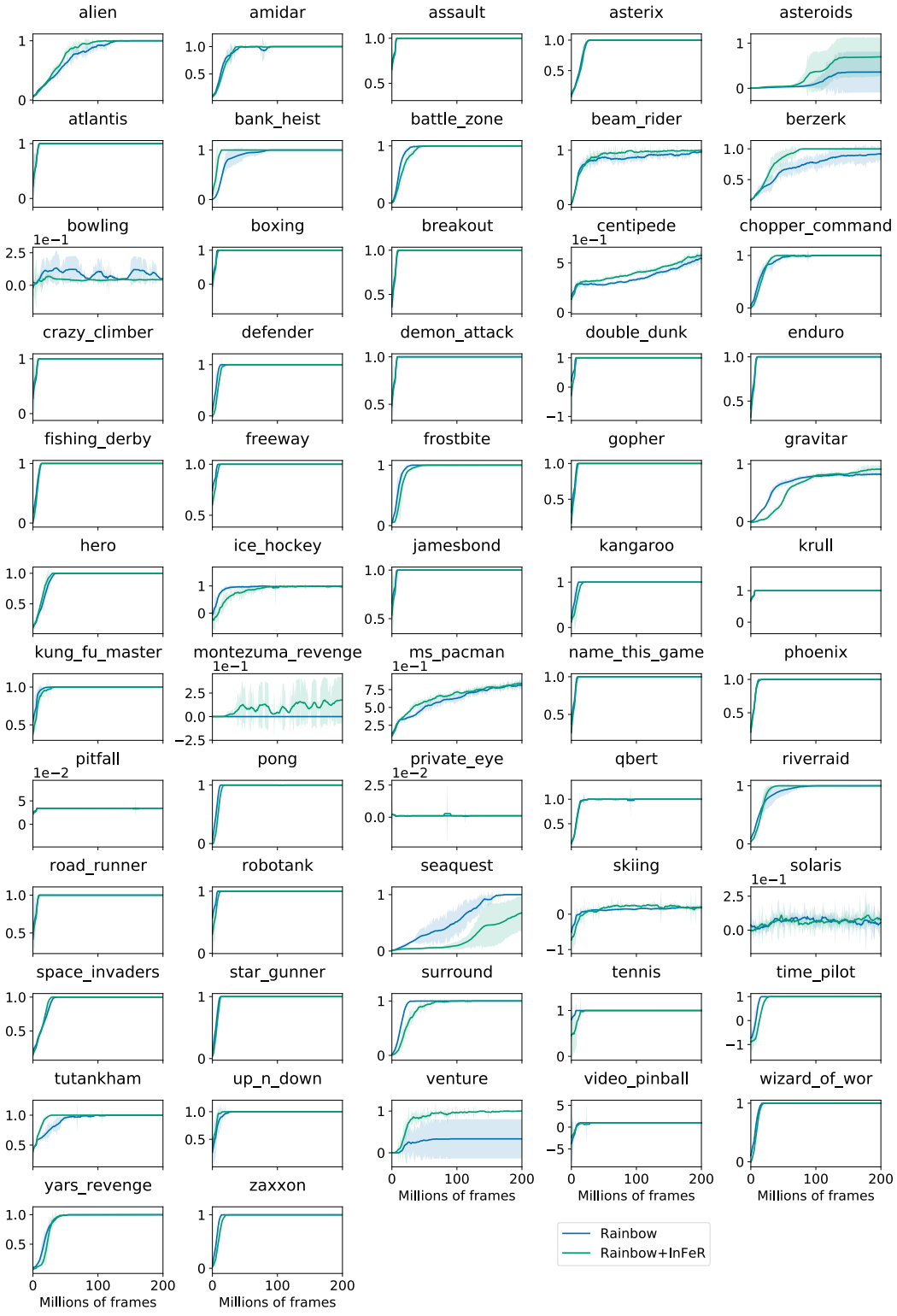

Figure 13: Full evaluation of capped human-normalized performance on Atari benchmarks in the double-width Rainbow architecture.

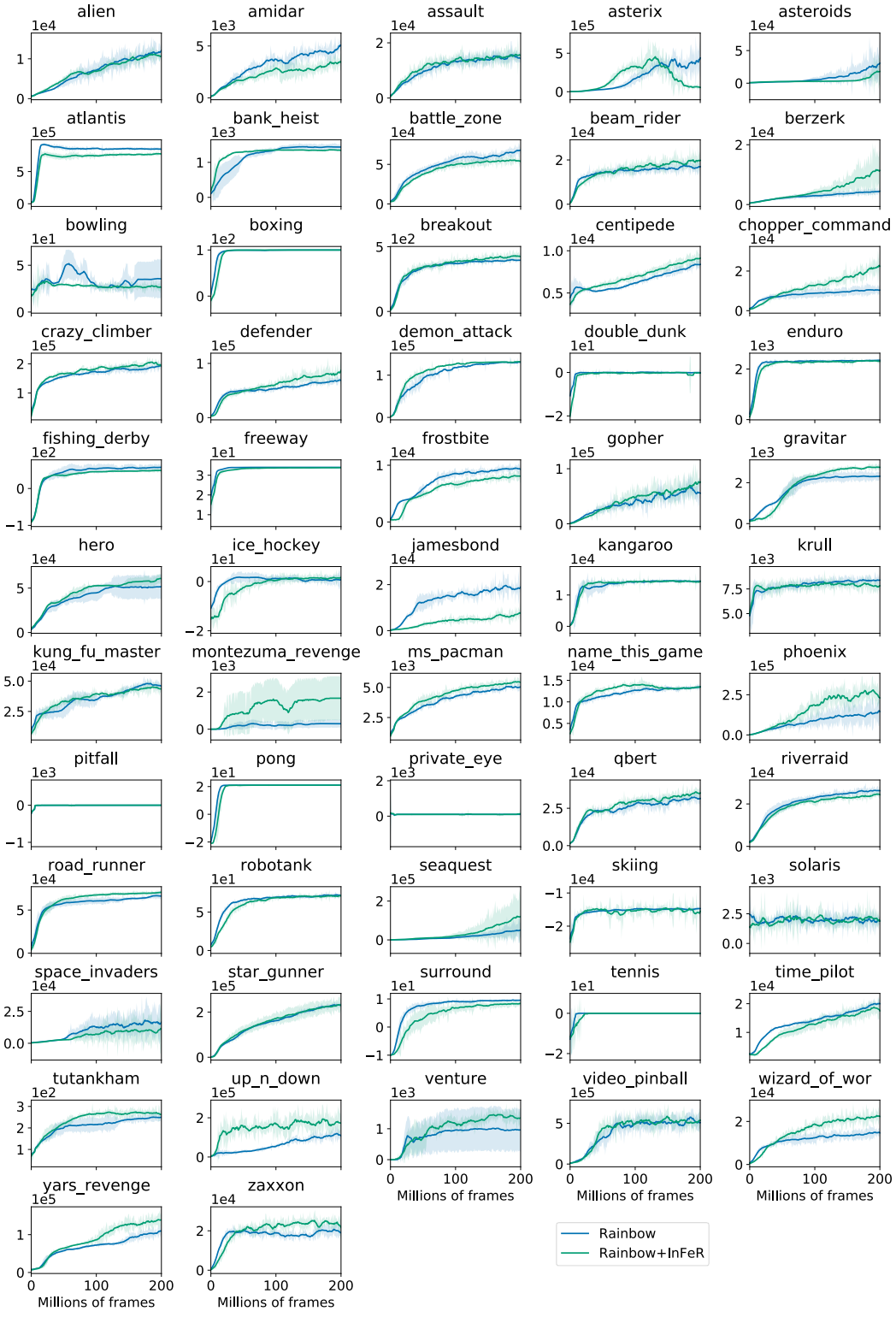

Figure 14: Full evaluation of raw scores on Atari benchmarks for the default Rainbow architecture.

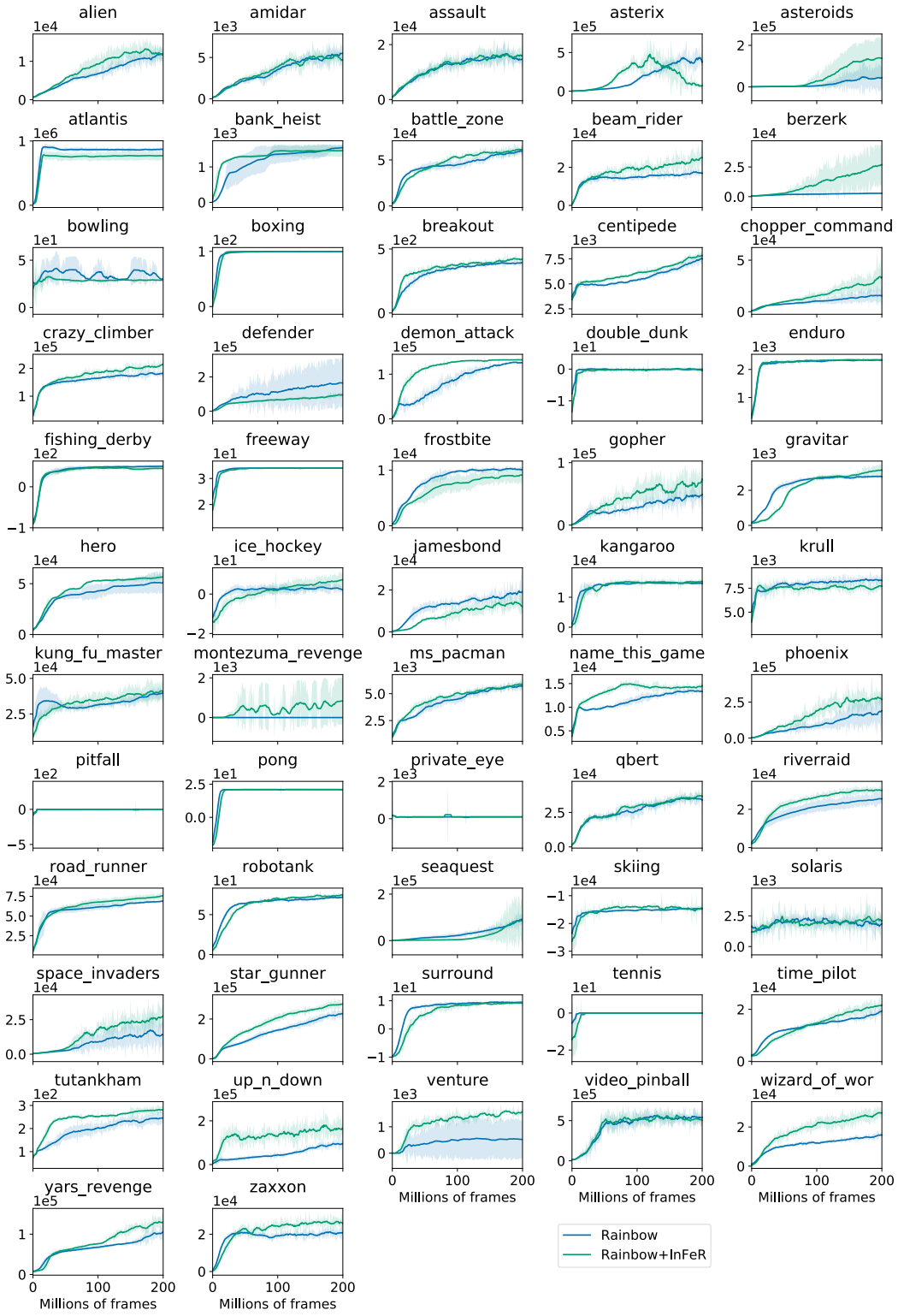

Figure 15: Full evaluation of raw scores on Atari benchmarks for the double-width Rainbow architecture.

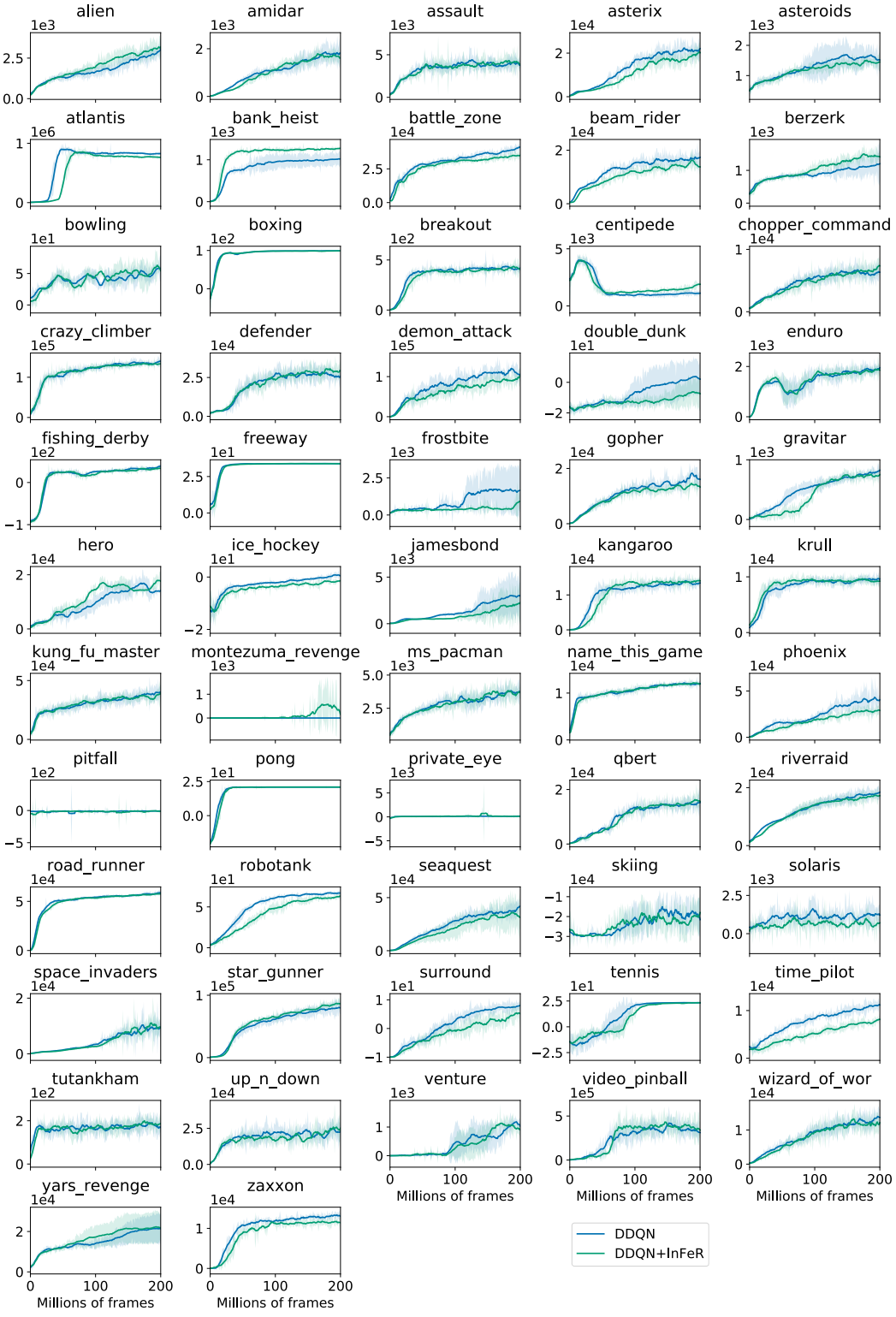

Figure 16: Evaluations of the effect of InFeR on performance of a Double DQN agent. Overall we do not see as pronounced an improvement as in Rainbow, but note that the average human-normalized score over the entire benchmark is nonetheless slightly higher for the InFeR agent, and that the performance improvement obtained by InFeR in Montezuma's Revenge is still significant in this agent.

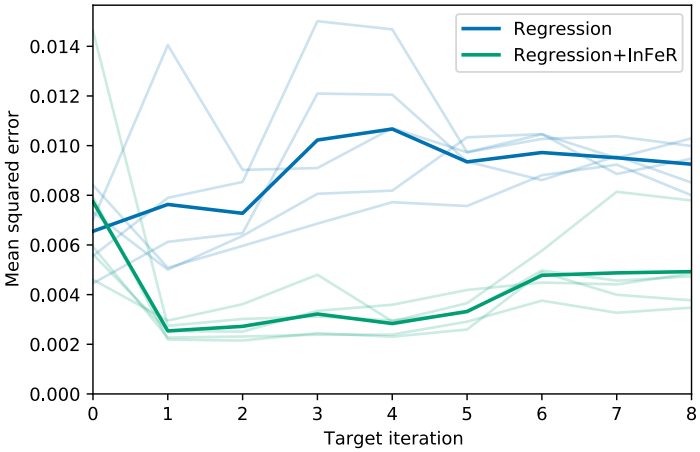

Figure 17: Effect of adding InFeR to the regression objective in a random reward prediction problem on the non-stationary MNIST environment. We see that the InFeR objective produces networks that can consistently outperform those trained with a standard regression objective, and even appears to enable forward-transfer early in training.