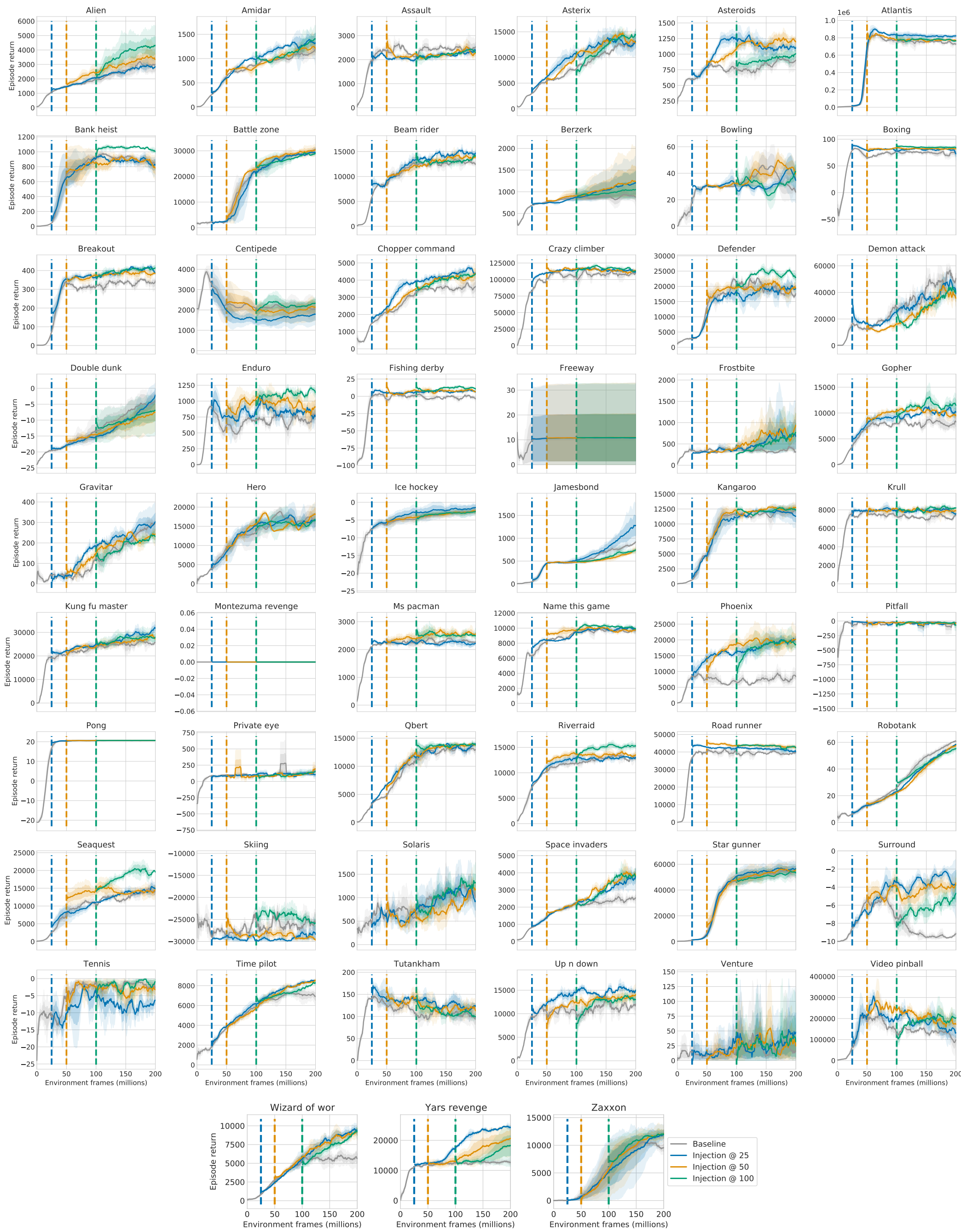

Figure 7: Performance of the Double DQN with and without plasticity injection after 25M, 50M, and 100M frames on the full Atari 57 benchmark. The potential discontinuities in the plots such as in `Road runner` are caused by the evaluation each 1M frames, i.e. the first moment the agent with injection contributes to the plot is after learning for 1M frames.

Injection Effect	Environments
Consistent Improvement	Alien, Asteroids, Breakout, Chopper command, Enduro, Frostbite, Gopher, Phoenix, Space invaders, Surround, Wizard of wor, Yars revenge (12 total)
Minor Improvement	Amidar, Asterix, Atlantis, Bank heist, Beam rider, Berzerk, Boxing, Defender, Fishing derby, Jamesbond, Krull, Ms pacman, Road runner, Seaquest, Time pilot, Up n down, Video pinball, Zaxxon (18 total)
Negligible	Battle zone, Bowling, Centipede, Crazy climber, Double dunk, Freeway, Gravitar, Hero, Ice hockey, Kangaroo, Kung fu master, Montezuma revenge, Name this game, Pitfall, Pong, Private eye, Qbert, Riverraid, Skiing, Solaris, Star gunner, Tennis, Tutankham, Venture (24 total)
Negative	Assault, Demon attack, Robotank (3 total)

Table 1: Summary of effects from applying plasticity injection to Double DQN on 57 Atari games.

A Complete Learning Curves

Figure 7 presents the return plots over the course of Double DQN training for 200M frames on the whole set of 57 Atari games. We informally categorized environments into four buckets upon visual inspection of effects from plasticity injection in Table 1. The most notable negative example is `Demon attack`, while on `Assault` and `Robotank` the effect is negative but minor. In the rest of the 54 games, plasticity injection either improves performance or has a negligible effect, possibly depending on the injection timestep.

B Ablations

This appendix presents an ablation analysis of the various design choices made during the study of plasticity injection. The purpose of such ablations is to build intuition on the behavior of plasticity injection under different conditions so that an RL practitioner can use it in their application.

Injection Variants. The proposed modification of the network architecture is not the only one possible. In Section 4, we initially described a version of plasticity injection without encoder sharing, that is, when the intervention is applied to the entire network (referred to as *Injection, Whole Net* in Figure 8). Another alternative is to create a whole new set of parameters and copy the encoder parameters of the old network without sharing it (denoted as *Injection, Whole Net, Copy Enc*). Lastly, for all three versions, there is the possibility of *not* freezing the old set of parameters (weights corresponding to the third, output correction term are always going to be frozen).

Figure 8 (left) summarizes the findings:

1. Creating a completely new encoder-head pair is the alternative with the lowest IQM scores;
2. Variants with encoder sharing or copying have comparable performance; the *Injection, Whole Net, Copy Enc* version has a slightly lower performance than the rest. We conjecture that it might be due to the larger number of frozen parameters;
3. Unfrozen variants generally perform not worse than their frozen counterparts. The unfrozen variants introduce more trainable parameters compared to the baseline, which require more computations during learning and increase the network expressivity. Since we were interested in a careful diagnosis of plasticity loss and extra expressivity may be a confounding factor, we decided to stick to the frozen version by default.

Multiple Injections. Given the improved performance from plasticity injection in the previous experiments, a natural question is whether applying plasticity injection multiple times would improve performance even further. To investigate this question, we applied plasticity injection at 100M and 150M frames, in addition to 50M frames, and plotted the IQM improvements with respect to a single injection at 50M frames. As shown in Figure 8 (right), additional injections do not improve the performance over a single injection in a setup with a standard network. We hypothesize that in our particular experimental setting, loss of plasticity can be largely mitigated with a single plasticity

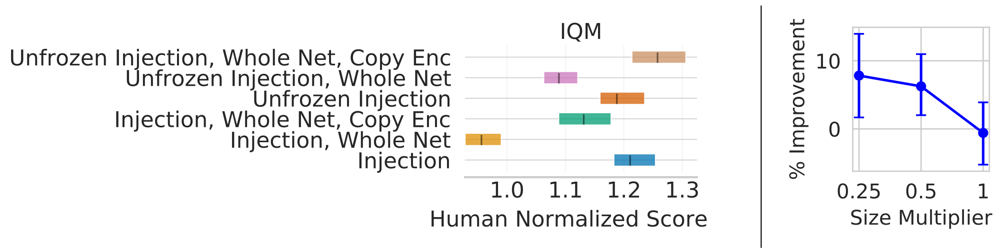

Figure 8: **Left:** Comparison between variations of plasticity injection. *Whole Net* denotes injection of both the encoder $\phi(\cdot)$ and the head $h_\theta(x)$; *Copy Enc* denotes copying the $\phi(\cdot)$ at the moment of injection without further sharing; *Unfrozen* denotes keeping parameters of the first term unfrozen. Relying on a new encoder leads to a lower performance; the rest of the alternatives have comparable scores. **Right:** Percentage improvements of the IQM score from multiple injections over a single injection for varying network sizes. Multiple injections are beneficial for smaller networks. Note that previous plots in Figure 5 show improvements when comparing one injection over no injections while this plot compares multiple injections over one.

injection. To verify this hypothesis, we applied multiple injections while varying the network size (similarly to Section 5.2, to make the network 2x smaller, we divide the width of the hidden layers by $\sqrt{2}$). Figure 8 (right) confirms that the level of improvement grows monotonically as the agent uses smaller networks. Since the results in Figure 5 suggests that the degree of plasticity loss increases with smaller networks, this result indicates that multiple rounds of plasticity injection can be beneficial in situations where the agent network is too small to maintain plasticity.

No Output Correction. In the majority of the games, subtracting the initial copy of the newly introduced head $h_{\theta_2}(\cdot)$ resulted in mostly similar learning curves as without the subtraction, although not always. In particular, the impact of the injection on `Yars Revenge` is smaller without compensating for the bias. Also, we observed a significant difference in high variance games (such as `Berzerk` and `Hero`). Note that removing effects on the predictions from introducing the new head would be possible by modifying the initialization [Brohan et al., 2022]. From the analysis viewpoint, we strove to have as clean experimental design as possible and wanted to remove initialization-specific confounders since initialization would affect network plasticity as well [Sutskever et al., 2013]. From the saving memory and computations viewpoint, it might be preferrable to do plasticity injection without introducing the third network.

Optimizer. One might hypothesize that benefits from injection can be attributed to manipulations with the optimizer state. To test this hypothesis, we perform two ablations: the first resets statistics of the RMSProp optimizer [Tieleman et al., 2012] used by Double DQN after 50M steps, the second copies the optimizer state of the original head to the newly initialized head after the injection. Figure 9 (left) demonstrates that most of the effects from injection come from having additional weights rather than from interventions on the optimizer.

Injection Timestep. In Section 5.2, we presented the results for a selection of environments while varying injection timestep. Figure 9 (right) suggests that across all games, changing the timestep by a

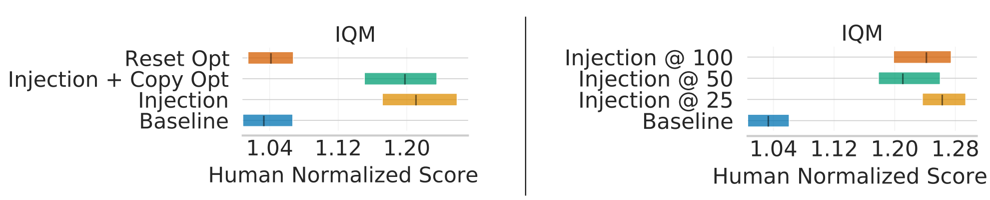

Figure 9: **Left:** Comparison of an agent with injection, an agent with injection but copied optimizer state for the newly initialized head (*Injection + Copy Opt*), and an agent that resets the optimizer statistics of the last two layers (*Reset Opt*). The results suggest that effects from interventions on the optimizer state are marginal compared to having new weights. **Right:** Aggregate performance for agents with varying injection timesteps. Whilst Figures 7 and 10 suggest that loss of plasticity might be happening at different paces across environments, the final IQM score is relatively robust with respect to the injection moment.

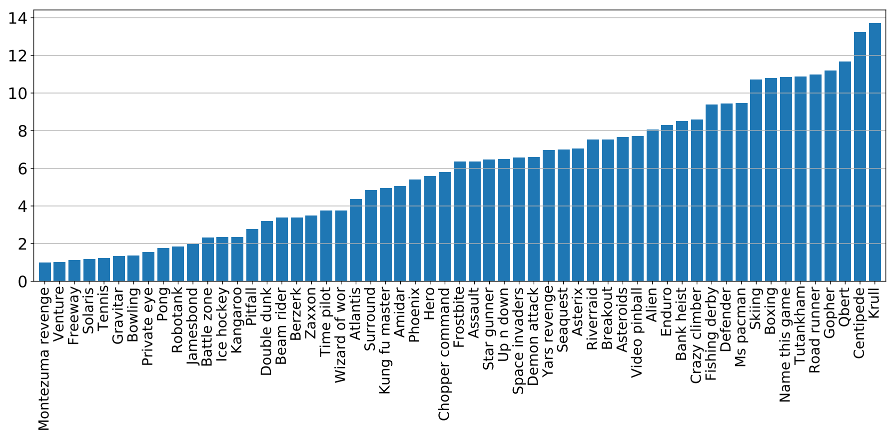

Figure 10: Per-game ratios of weight magnitude after learning for 200M frames and before experiencing any data. The ratios can vary up to 10 times between games.

factor of two yields comparable aggregate performance. Note though that we measure the IQM score after 200M frames, so the transient performance would differ depending on the timestep.

Adaptive Criterion for Injection. As a step towards getting rid of the need to specify the injection timestep, we also explored the option of having a criterion for triggering the intervention. If the agent has the initial weight magnitude $\|w_0\|$ (w denotes here both encoder and head weights), we inject plasticity after the weight norm surpasses the $3\|w_0\|$ threshold. The IQM scores of the agent with injection after 50M steps and with this heuristic coincide, although the frame when the agent reaches the threshold differs per game significantly: for some environments, it can be as small as 20M (such as `Enduro`), for other environments, it can be beyond 200M (such as `Robotank`) implying that the agent will learn without injection. Figure 10 gives an overview of how much the weight norm grows over the course of training (suggesting how fast the agent reaches the $3\|w_0\|$ threshold on each game). We view devising an even more powerful criterion as a promising avenue for future work.

L2 Regularization. The observations about the norm increase made us try adding L2 regularization to the Double DQN agent. A grid search over $[10^{-7}, 3 \cdot 10^{-7}, 10^{-6}, 3 \cdot 10^{-6}, 10^{-5}, 3 \cdot 10^{-5}]$ coefficients resulted in the best coefficient of $3 \cdot 10^{-6}$ but leaving the aggregate score mostly the same; higher values resulted in significant performance deterioration. The result gives evidence that controlling the weight norm itself does not address plasticity loss but allows multiple interpretations. We speculate that L2 might be prematurely encouraging weights to have zero magnitude before obtaining high rewards (the effect would be especially profound in sparse reward settings) or that L2 might have undesirable side effects of smoothing approximate value functions while the true value functions might be non-smooth [Dong et al., 2020]. We are puzzled about the inefficacy of L2 in our experiments and mixed results from applying it in RL in past works: the majority of deep RL algorithms do not use it [Mnih et al., 2015, Schulman et al., 2015, Lillicrap et al., 2015, Mnih et al., 2016, Bellemare et al., 2017], although not without exceptions [Schrittwieser et al., 2021]. Some works have explicitly reported negative effects from controlling the weight norm in deep RL [Nikishin et al., 2022], while others highlighted its benefits [Farahmand et al., 2008, Li et al., 2023]; more research in needed to understand its effect in RL.

C Details about the Baselines

In Section 5.3, we considered three alternative ways of dynamically addressing plasticity loss during training: resets, Shrink-and-Perturb (SnP), and naive width scaling. Resets re-initialize parameters of the last layers (using our notation, it corresponds to replacing $h_\theta(\cdot)$ with $h_{\theta'_1}(\cdot)$)) and rely on a replay buffer to transfer knowledge before and after the intervention. Resets require the specification of the number of last layers and the application timestep. We ran a sweep over [1, 2] layers and two choices of timesteps: either once at 50M frames or trice at 50M, 100M, and 150M. Afterwards, we reported the results that attain the highest IQM score.

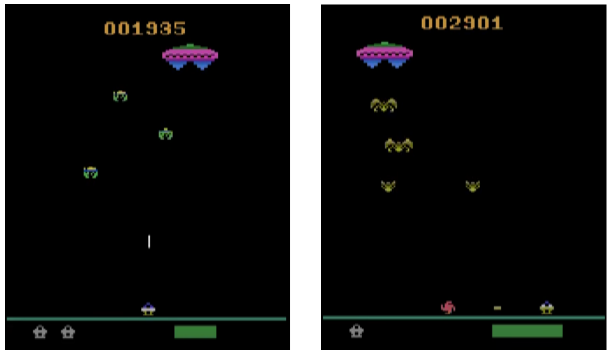

Figure 11: A demonstration of the `Assault` game evolution when a high-performing agent found on the Internet reaches a score of around 2800: before, the agent had to shoot only upwards; afterwards, it has to shoot up, left, and right. We interpret that the failure to improve upon the 2800 score is explained by exploration.

Shrink-and-Perturb modify all network weights w as $w \leftarrow \lambda w + \sigma \epsilon$ at the given application timesteps, where ϵ is a random vector with the same dimensionality as w sampled from the standard Gaussian distribution. SnP has three hyperparameters: the shrink coefficient λ, the noise scale σ, and the application timesteps. We performed a grid search over λ in [0.1, 0.3, 1], σ in [0.01, 0.1, 1], and the same choices of timesteps as for resets.

The best hyperparameters ended up being the ones that somewhat minimized the effect of both resets (1 layer, 1 application time) and SnP ($\lambda = 1$, $\sigma = 0.01$, 3 application times); other hyperparameters resulted in even worse performance. The paper on resets [Nikishin et al., 2022] demonstrates results on the Atari 100k benchmark [Kaiser et al., 2019] that focuses on a data-efficient regime with 10^5 interactions only and contains a subset of 26 (out of 57) games. In this setting, the replay buffer has all experiences encountered during the agent's lifetime; this data can be sufficient for recovering the performance after a reset. In the Atari 200M setting though, the replay buffer has only 4M frames which might not be enough to recover fast after a reset. We speculate that similar reasoning applies to SnP since it can be seen as a soft version of resets [D'Oro et al., 2023].

For the width scaling method, we modify the last two layers by doubling their width. In detail, suppose the weight matrices are $W_1 \in \mathbb{R}^{N \times K}$ and $W_2 \in \mathbb{R}^{K \times |\mathcal{A}|}$, where $|\mathcal{A}|$ is the action space dimensionality. We create two new matrices $W_1' \in \mathbb{R}^{N \times 2K}$ and $W_2' \in \mathbb{R}^{2K \times |\mathcal{A}|}$ and fill the first K columns of W_1' with values of W_1 and the first K rows of W_2' with values of W_2. The remaining entries are sampled from the random initializer. We perform a modification to the bias term $b_1' \in \mathbb{R}^{2K}$ by copying values from $b_1 \in \mathbb{R}^K$ and setting the rest to zero. The width is scaled once at 50M.

Such a naive approach increases plasticity but its inability to improve over the standard Double DQN might be caused by adverse effects on the predictions after the intervention without output correction.

D The `Assault` Game Analysis

We searched for a high-scoring behavior demonstration in the `Assault` environment on YouTube[3]. The screenshots in Figure 11 demonstrate the change of the environment around the score of 2800: before, the enemies were appearing only above the controlled starship, while afterwards, they start to appear from the left and from the right. Before the transition, the algorithm learned that actions "shoot left" and "shoot right" were irrelevant, while afterwards, it has to start using these actions, suggesting that the performance plateau can be attributed to exploration challenges.

We highlight that it was the suggested protocol for diagnosis that led to the insight: after seeing that the post-injection agent has the same performance plateau as the baseline, we decided to investigate the behavior in the game and realized that previously irrelevant actions became critical.

[3]https://youtu.be/HwWJrb2PQQ0