# ADAPTIVE RATIONAL ACTIVATIONS TO BOOST DEEP REINFORCEMENT LEARNING

**Quentin Delfosse**[*,1], **Patrick Schramowski**[1,2,3], **Martin Mundt**[1,3], **Alejandro Molina**[1] & **Kristian Kersting**[1,2,3,4]

[1] Computer Science Dept., TU Darmstadt
[2] German Center for Artificial Intelligence
[3] Hessian Center for Artifical Intelligence
[4] Centre for Cognitive Science, Darmstadt

## ABSTRACT

Latest insights from biology show that intelligence not only emerges from the connections between neurons, but that individual neurons shoulder more computational responsibility than previously anticipated. Specifically, neural plasticity should be critical in the context of constantly changing reinforcement learning (RL) environments, yet current approaches still primarily employ static activation functions. In this work, we motivate the use of adaptable activation functions in RL and show that rational activation functions are particularly suitable for augmenting plasticity. Inspired by residual networks, we derive a condition under which rational units are closed under residual connections and formulate a naturally regularised version. The proposed joint-rational activation allows for desirable degrees of flexibility, yet regularises plasticity to an extent that avoids overfitting by leveraging a mutual set of activation function parameters across layers. We demonstrate that equipping popular algorithms with (joint) rational activations leads to consistent improvements on different games from the Atari Learning Environment benchmark, notably making DQN competitive to DDQN and Rainbow.[1]

## 1 INTRODUCTION

Neural Networks' efficiency in approximating any function has made them the default choice in many machine learning tasks. This is no different in deep reinforcement learning (RL), where the DQN algorithm's introduction (Mnih et al., 2015) has sparked the development of various neural solutions. In concurrence with former neuroscientific explanations of brainpower residing in combinations stemming from trillions of connections (Garlick, 2002), present advances have emphasised the role of the neural architecture (Liu et al., 2018; Xie et al., 2019). As such, RL improvements have first been mainly obtained through enhancing algorithms (Mnih et al., 2016; Haarnoja et al., 2018; Banerjee et al., 2021) and only recently by searching for performing architectural patterns, via automatic deep policy search (Pang et al., 2021; Krishnan et al., 2023), or via decoupling object detection (Lin et al., 2020; Delfosse et al., 2023b) and policy search (Delfosse et al., 2023a; Wu et al., 2024).

However, research has also progressively shown that individual neurons shoulder more complexity than initially expected, with the latest results demonstrating that dendritic compartments can compute complex functions (*e.g.* XOR) (Gidon et al., 2020), previously categorised as unsolvable by single-neuron systems. This finding seems to have renewed interest in activation functions (Georgescu et al., 2020; Misra, 2020). In fact, many functions have been adopted across different domains (Redmon et al., 2016; Brown et al., 2020; Schulman et al., 2017). To reduce the bias introduced by a fixed activation function and achieve higher expressive power, one can further learn which activation function is performant for a particular task (Zoph & Le, 2017; Liu et al., 2018), learn to combine arbitrary families of activation functions (Manessi & Rozza, 2018), or find coefficients for polynomial activations as weights to be optimised (Goyal et al., 2019).

Whereas these prior approaches have all contributed to their respective investigated scenarios, there exists a finer approach that elegantly encapsulates the challenges brought on by reinforcement

[1] Rational library: github.com/k4ntz/activation-functions; Experiments: github.com/ml-research/rational_rl.

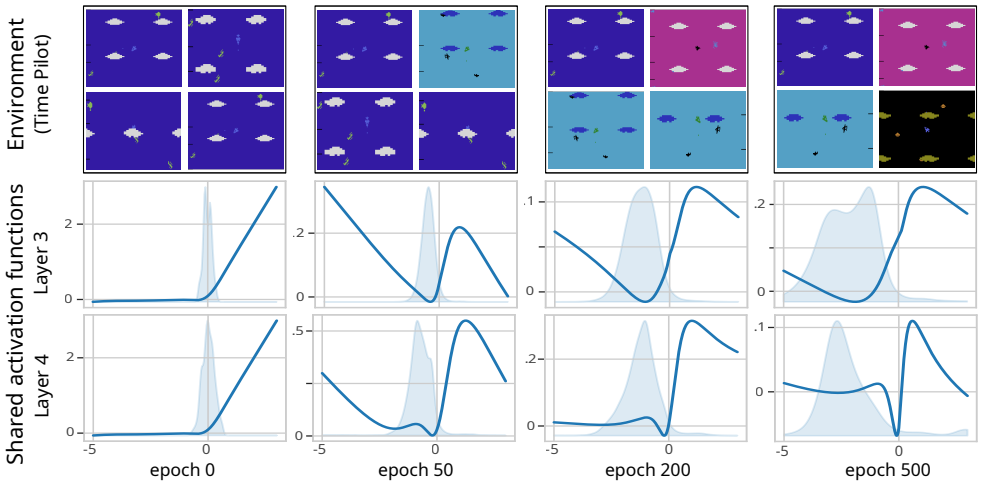



Figure 1: **Neural plasticity due to trainable activation functions allows RL agents to adapt to environments of increasing complexity.** Rational activations (bottom), with shared parameters in each of the last two layers, evolve together with their input distributions (shaded blue) when learning with DQN on Time Pilot. Each column corresponds to a training state where a new, more challenging part of the environment (top, e.g. increasing enemy speed and movement complexity) has been uncovered and is additionally used for training.

learning problems. Specifically, at each layer, we can learn rational activation functions (ratio of polynomials) (Molina et al., 2020). Not only can rationals converge to any continuous function, but they have further been proven to be better approximants than polynomials in terms of convergence (Telgarsky, 2017). Even more crucially, their ability to adapt while learning equips a model with high *neural plasticity* ( "capability to adjust to the environment and its transformations" (Garlick, 2002)). We argue that adapting to environmental changes is essential, making rational activation functions particularly suitable for dynamic RL environments. To provide a visual intuition, we showcase an exemplary evolution of two rational activation functions together with their respective changing input distributions in the dynamic "Time Pilot" environment in Fig. 1.

In this work, we **show that plasticity is of major importance for RL agents**, as a central element to satisfy the requirements originating from diverse and dynamic environments and **propose the use of rational activation functions to augment deep RL agents plasticity**. Apart from demonstrating the suitability of adaptive activation functions for Deep RL, we also evaluate how many additional layer weights can be replaced by rational activations. Our specific contributions are:

**(i)** We motivate why neural plasticity is a key aspect for Deep RL agents and that rational activations are adequate as adaptable activation functions. For this purpose, we not only highlight that rational activation functions adapt their parameters over time, but further prove that they can dynamically embed residual connections, which we refer to as residual plasticity.

**(ii)** As additional representational capacity can hinder generalisation, especially in RL (Farebrother et al., 2018; Roy et al., 2020; Yarats et al., 2021), we propose a joint-rational variant, that uses weight-sharing in rational activations across different layers.

**(iii)** We empirically demonstrate that rational activations bring significant improvements to DQN and Rainbow algorithms on Atari games and that our joint variant further increases performance.

**(iv)** Finally, we investigate the overestimation phenomenon of predicting too large return values, which has previously been argued to originate from an unsuitable representational capacity of the learning architecture (van Hasselt et al., 2016). As a result of our introduced (rational) neural and residual plasticity, such overestimation can practically be reduced.

We proceed as follows. We start off by arguing in favour of plasticity for deep RL, then show how rational functions are particularly suitable candidates to provide plasticity in neural networks and present our empirical evaluation. Before concluding, we touch upon related work.

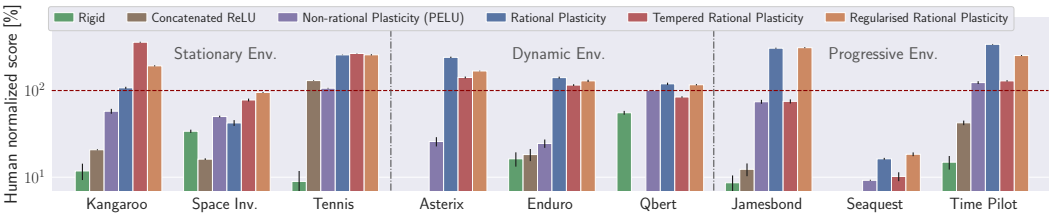



Figure 2: **Neural plasticity is essential for reinforcement learning**. Human normalised mean scores for rigid (LReLU and CRELU) DQN agents, agents with non-rational, rational, tempered, and regularised plasticity are shown with standard deviation across 5 random seeded experimental repetitions. Larger scores are better. Tempered plasticity, allowing initial adaptation to the environments, but not their transformations in experimental repetitions, performs better on stationary environments. Regularised plasticity performs well across all environment types. Best viewed in colour. A description of the environments' types is provided in Appendix A.5.

## 2 Rational Plasticity for Deep RL

Let us start by arguing why deep reinforcement learning agents require extensive plasticity and show that parametric rational activation functions provide appropriate means to augment plasticity.

As motivated in the introduction, RL is subject to inherent distribution shifts. During training, agents progressively uncover new states (input drift) and, as the policy improves, the reward signal is modified (output drift). More precisely, for input drifts, we can distinguish environments according to how much they change through learning. For simplicity, we categorise according to three intuitive categories: stationary, dynamic and progressive environments. Consider the example of Atari 2600 games. Kangaroo and Tennis can be characterised as stationary since the games' distributions do not change significantly through learning. Asterix, Enduro and Q*bert are dynamic environments, as different distribution shifts (*e.g.* Cauldron, Helmet, Shield, etc. in Asterix) are provided to the agents in the first epochs, with no policy improvement required to uncover them. On the contrary, Jamesbond, Seaquest, and Time Pilot are progressive environments: agents need to master early stages before being provided with additional states, *i.e.* exposed to significant input shifts.
How do we efficiently improve RL agents' ability to adapt to environments and their changes? To deal with distribution shifts, our agents require high neural plasticity and thus benefit from adaptive architectures. To elaborate further in our work, let us consider the popular DQN algorithm (Mnih et al., 2015), that employs a $\boldsymbol{\theta}$-parameterised neural network to approximate the Q-value function of a state $S_t$ and action $a$. This network is updated following the Q-learning equation: $Q\left(S_t, a; \boldsymbol{\theta}\right) \equiv R_{t+1} + \gamma \max_a Q\left(S_{t+1}, a; \boldsymbol{\theta}\right)$. In addition to network connectivity playing an important role, we now highlight the importance of individual neurons by modifying the network architecture of the algorithm via the use of learnable activation functions, to show that they are a presently underestimated component. To emphasise the utility of the upcoming proposed rational and joint-rational activation functions, we will interleave early results into this section. The latter serves the primary purpose to not only motivate the suitability of the rational parameterisation to provide plasticity, but also discern the individual benefits of (joint-) rational activations, in the spirit of ablation studies.

### 2.1 Rational Neural Plasticity

Rational functions are ratio of polynomials, defined on $\mathbb{R}$ by $\mathrm{R}(x) = \dfrac{\mathrm{P}(x)}{\mathrm{Q}(x)} = \dfrac{\sum_{j=0}^{m} a_j x^j}{1 + \sum_{k=1}^{n} b_k x^k}$, where $x \in \mathbb{R}$, $\{a_j\}$ and $\{b_k\}$ are $m+1$ and $n$ (real) learnable parameters per layer. To test rational functions' plasticity, we use the discrete distribution shifts of the Permutted-MNIST continual learning experiment. We show in Appendix 4 that rational activation functions improve the plasticity over both ReLU and over CReLU (Shang et al., 2016), used to augment plasticity by Abbas et al. (2023). For RL we show in Fig. 2 that the rational parametrisations substantially enhances RL agents. More precisely, by comparing agents with rigid networks (a fixed Leaky ReLU baseline) to agents with rational plasticity (*i.e.* with a rational activation function at each layer), we see that rational functions boosts the agents to super-human performances on 7 out of 9 games. The acquired extra neural plasticity seems to play a significant role in these Atari environments, especially in progressive ones.

In order to discern the benefits of general plasticity through any adaptive activation function, over the proposed use of the rational parametrisation, Fig. 2 additionally includes agents with Concatenated ReLU and with Parametrised Exponential Linear Unit (Trottier et al., 2017). CReLU, was used in RL to address plasicity loss (Abbas et al., 2023), outperforms LRELU on 6 out of 9 games, but is always outmatched by rational plasticities. PELU that uses 3 parameters to control its slope, saturation and exponential decay, and has been shown to outperform other learnable alternatives on classification tasks (Godfrey, 2019). However, in contrast to the rational parameterisation, it seems to fall behind and only boosts the agents to super-human performance on 3 out of 9 games (contrary to 7), implying that the type of plasticity provided by rational activations is particularly suitable.

To highlight the desirability of rational activations even further, we additionally distinguish between the plasticity of agents towards their specific environment and the plasticity allowing them to adapt while the environment is changing. To this end, we show agents equipped with rational activations that are tempered in Fig. 2. Such agents are equipped with the final, optimised rational functions of trained rational-equipped agents. They correspond to frozen functions from agents that already adapted to their specific environment. The plasticity of the rationals is thus tempered (*i.e.* stopped) in a repeated application (*i.e.* another training session) to emphasise the necessity to continuously adapt the activation functions, together with the layers' weights during training. Whereas providing agents with such tempered, tailored to the task, activations already boosts performances, rational plasticity at all times is essential, particularly in dynamic and progressive environments.

### 2.2 Rational Residual Plasticity

The prior paragraphs have showcased the advantage of agents equipped with rational activation functions, rooted in their ability to update their parameters over time. However, we argue that the observed boost in performance is not only due to parameters adapting to distributional drifts. Rational activations can embed one of the most popular techniques to stabilise deep networks training; namely, they can dynamically make use of a residual connection. We refer to this as residual plasticity.

**Rationals are closed under residual connection and provide residual plasticity.** We here show that rational functions with strictly higher degree in the numerator embed residual connections. Recall that residual neural networks (ResNets) were initially introduced following the intuition that it is easier to optimise the residual mapping than to optimise the original, unreferenced mapping (He et al., 2016). Formally, residual blocks of ResNets propagate an input $X$ through two paths: a transforming block of layers that preserves the dimensionality ($F$) and a residual connection (identity).

**Theorem:** Let R be a rational function of order (m, n). R embeds a residual connection $\Leftrightarrow m > n$.

*Proof:* Let us consider a rational function R = P/Q of order $(m, n)$, with coefficients $A^{[m]} = (a_j)_{j=0}^{m} \in \mathbb{R}^{m+1}$ of P and $B^{[n]} = (b_i)_{i=0}^{n} \in \mathbb{R}^{n+1}$ of Q (with $b_0 = 1$). We denote by $\otimes$ (respectively $\oslash$) the Hadamard product (respectively division). Let $X \in \mathbb{R}^{n_1 \times \cdots \times n_x}$ be a tensor corresponding to the input of the rational function of an arbitrary layer in a given neural network. We derive $X^{\otimes k} = \bigotimes_{i=1}^{k} X$. Furthermore, we use $GV^{[k]}(X) = [\mathbf{1}, X, X^{\otimes 2}, ..., X^{\otimes k}] \in \mathbb{R}^{(n_1 \times \cdots \times n_x) \times k+1}$ to denote the tensor containing the powers up to $k$ of the tensor $X$. Note that $GV^{[k]}$ can be understood as a generalised Vandermonde tensor, similar as introduced in (Xu et al., 2016). For $V^{[k]} = (v_i)_{i=0}^{k} \in \mathbb{R}^{k+1}$, let $GV^{[k]}.V^{[k]} = \sum_{i=0}^{k} v_i X^{\otimes i}$ be the weighted sum over the tensor elements of the last dimension.

Now, we apply the rational activation function R with residual connection to $X$:

$$
\begin{aligned}
\mathbf{y}(X) = \mathrm{R}(X) + X &= GV^{[m]}(X).A^{[m]} \oslash GV^{[n]}(X).B^{[n]} + X \\
&= (GV^{[m]}(X).A^{[m]} + X \otimes GV^{[n]}(X).B^{[n]}) \oslash GV^{[n]}(X).B^{[n]} \\
&= (GV^{[m]}(X).A^{[m]} + GV^{[n+1]}(X).B_0^{[n+1]}) \oslash GV^{[n]}(X).B^{[n]} \\
&= GV^{[\max(m,n+1)]}(X).C^{[\max(m,n+1)]} \oslash GV^{[n]}(X).B^{[n]} \qquad = \widetilde{\mathrm{R}}(X),
\end{aligned}
$$

where $B_0^{[n+1]} = (b_{0,i})_{i=0}^{n+1} \in \mathbb{R}^{n+2}$ (with $b_{0,0} = 0$ and $b_{0,i} = b_{i-1}$ for $i \in \{1, ..., n+1\}$),
$C^{[\max(m,n+1)]} = (c_j)_{j=0}^{\max(m,n+1)}$ ($c_j = a_j + b_{j-1}$, $a_j = 0 \; \forall j \notin \{0, ..., m\}$, $b_j = 0 \; \forall j \notin \{0, ..., n\}$).
$\widetilde{\mathrm{R}}$ is a rational function of order $(m', n')$, with $m' > n'$. In other words, rational activation functions of order $m > n$ embed residual connections. Using the same degrees for numerator and denominator

certifies asymptotic stability, but our derived configuration allows rationals to implicitly use residual connections. Importantly, note that these residual connections are not rigid, as these functions can progressively learn $a_j = 0$ for all $j > n$, *i.e.* we have residual plasticity.

**Rational plasticity to replace residual blocks' plasticity.** In very deep ResNets, it has been observed that feature re-combinations does not occur inside the blocks but that transitions to representations occur during dimensionality changes Veit et al. (2016); Greff et al. (2017).
To investigate this hypothesis, Veit et al. have conducted lesioning experiments, where a residual block is removed from the network, and surrounding ones are fine-tuned to recover. Whereas we emphasize that we do not claim that the residual in rationals can replace entire convolutional blocks or that they are generally equivalent, we hypothesize that under the conditions investigated by Veit et al. of very deep networks, residual blocks could learn complex activation function-like behaviours. To test this conjecture, we repeat the lesioning experiments, but also test replacing the lesioned block with a rational function that satisfies the residual connection condition derived above. Results are provided in appendix (*cf.* A.2) and show that **rational functions' plasticity can efficiently compensate for the lost capacity of lesioned residual blocks in very deep residual networks**.

### 2.3 Natural Rational Regularisation

We have motivated and shown that the combination of neural and residual plasticity form the central pillars for why rational activation functions are desirable in deep RL. In particular for dynamic and progressive environments, rational plasticity has been observed to provide a substantial boost over alternatives. However, if we circle back to Fig. 2 and take a more careful look at the stationary environments, we can observe that our previously investigated tempered rational plasticity (for emphasis, initially allowed to tailor to the task but later "stopped" in experimental repetition) can also have an upper edge over full plasticity. The extra rational plasticity at all times might reduce generalisation abilities, particularly on non-diverse stationary environments. In fact, prior works have highlighted the necessity for regularisation methods in deep reinforcement learning (Farebrother et al., 2018; Roy et al., 2020; Yarats et al., 2021).

We thus propose a naturally regularised rational activation version, inspired from residual blocks. In particular, Greff et al. have indicated that sharing the weights can improve learning performances, as shown in Highway (Lu & Renals, 2016) and Residual Networks (Liao & Poggio, 2016). In the spirit of these findings, we propose the regularised *joint-rationals*, where we constrain the input to propagate through different layers but always be activated by the same learnable rational activation function. Rational functions thus share a mutual set of parameters across the network, (instead of layers, *cf.* Fig. 11). As observable in Fig. 2, this regularised form of plasticity increases the agents' scores in the stationary environments and does not deteriorate performances in the progressive ones.

## 3 Empirical Evidence for Plasticity

Our intention here is to investigate the benefits of neural plasticity through rational networks for deep reinforcement learning. That is, we investigated the following questions:

**(Q1)** Do neural networks equipped with rational plasticity outperform rigid baselines?

**(Q2)** Can neural plasticity make up for more heavy algorithmic RL advancements?

**(Q3)** Can plasticity address the overestimation problem?

**(Q4)** How many more parameters would rigid networks need to measure up to rational ones?

To this end, we compare[2] our rational plasticity using the original DQN algorithm and its convolutional network (Mnih et al., 2015) on 15 different games of the Atari 2600 domain (Brockman et al., 2017). We compare these architectures to ones equipped with Leaky ReLU (as experiments on Breakout and SpaceInvaders showed that agents with Leaky ReLU outperform ReLU ones), the learnable PELU function, as well as SiLU ($\mathrm{SiLU}(x) = x \cdot \mathrm{sigmoid}(x)$) and its derivative dSiLU. Elfwing et al. showed that SiLU or a combination of SiLU (on convolutional layers) and its derivative (on fully connected layers) perform better than ReLU in DQN agents on several games (2018). SiLU and dSiLU are —to our knowledge— the only activation functions specifically designed for RL applications. More details on the architecture and hyperparameters can be found in Appendix A.8.

[2] 30.000 GPU hours, carried out on a DGX-2 Machine with Nvidia Tesla V100 with 32GB.

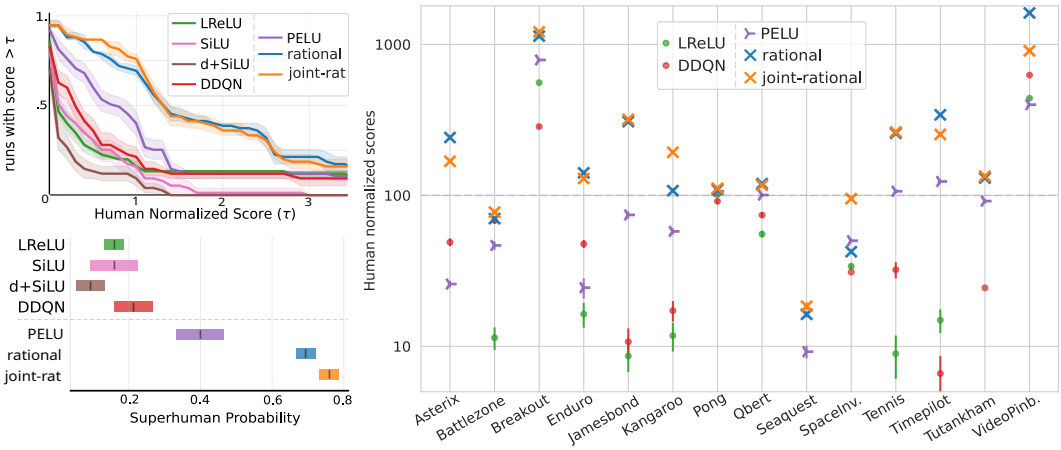



Figure 3: **Learnable functions' plasticity boosts RL agents.** For reliable evaluation, we report the performance profiles (top left) as well as superhuman probabilities (with CIs, bottom left) of baselines (*i.e.* DQN and DDQN with Leaky ReLU, DQN with SiLU and SiLU + dSiLU), as well as DQN with plasticity: using PELU, rational and joint-rational (5 random seeds). While the learnable PELU already augment performances of its agents, rational and joint-rational ones lift them above human performances on more than 70% of our runs. Detailed score tables are provided in Appendix A.3.

We then compare increased neural plasticity provided by (joint-)rational networks to algorithm improvements, namely the Double DQN (DDQN) method (van Hasselt et al., 2016), that tackles DQN's overestimation problem, as well as Rainbow (Hessel et al., 2018). Rainbow incorporates multiple algorithm improvements brought to DQN—Double Q-learning, prioritised experience replay, duelling network, multi-step target, distributional learning and stochastic networks—and is widely used also as a baseline (Lin et al., 2020; Hafner et al., 2021). We further explain how neural plasticity can help readdress overestimation. Finally, we evaluate how many additional weights rigid networks need to approximate rational ones.

In practice, we used safe rational functions (Molina et al., 2020), *i.e.* we used the absolute value of the sum in the denominator to avoid poles. This stabilises training and makes the function continuous. Rational activation functions are shared across layers (adding only 10 parameters per layer) or through the whole network for the regularised (joint) version, with their parameters optimised together with the rest of the weights. For ease of comparison and reproducibility, we conducted the original DQN experiment (also used by DDQN and SiLU authors) using the mushroomRL (D'Eramo et al., 2020) library, with the same hyperparameters (*cf.* Appendix A.8) across all the Atari agents, for a specific game, but we did not use reward clipping. For a fair comparison, we report final performances using the human-normalised (*cf.* Eq. 2 in Appendix) mean and standard deviation of the scores obtained by fully trained agents over five seeded reruns for every (D)DQN agent. However, since often only the best performing RL agent is reported in the literature, we also provide tables of such scores (*cf.* Appendix A.3). For the Rainbow algorithm, we unfortunately can only report the results of single runs. A single run took more than 40 days on an NVIDIA Tesla V100 GPU; Rainbow is computationally quite demanding (Obando-Ceron & Castro, 2021).

**(Q1) DQN with activation plasticity is better than rigid baselines.** To start off, we compared RL agents with additional plasticity (from PELU and rationals) to rigid DQN baselines: Leaky ReLU, as well as agents equipped with SiLU and SiLU+dSiLU activation functions.
The results summarised in Fig. 3 confirm what our earlier figure had shown, but on a larger scale. While RL agents with functions of the SiLU family do not outperform Leaky ReLU ones in our games, plastic DQN agents clearly outperform their rigid activation counterparts. DQN with regularised plasticity even obtains a higher superhuman probability and highest mean scores 64% of the time. Scores on (difficult credit assignment) Skiing are in Appendix A.3. This clearly shows that plasticity, and above all rational plasticity, pays off for deep agents, providing an affirmative answer to **Q1**.

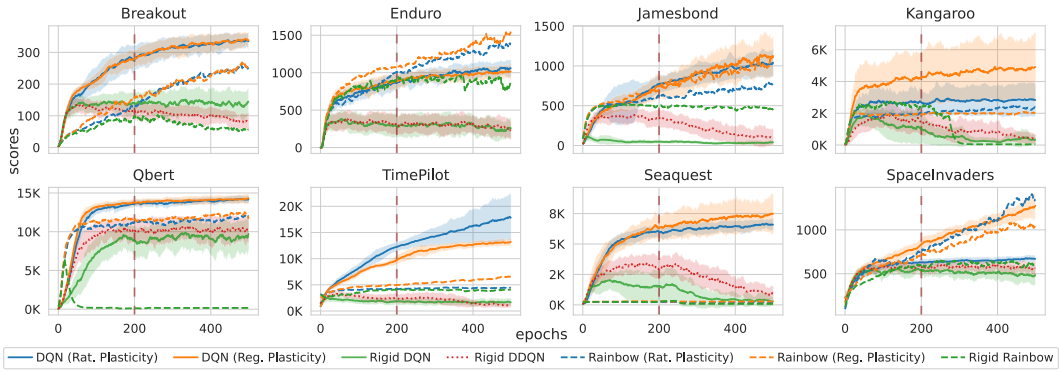



Figure 4: Networks with rational (Rat.) and regularised (Reg.) rational plasticity compared to rigid baselines (DQN, DDQN and Rainbow) over five random seeded runs on eight Atari 2600 games. The resulting mean scores (lines) and standard deviation (transparent area) during training are shown. As one can see, DDQN does not resolve performance drops but only delays them (e.g. particularly pronounced on Seaquest). A figure including the evolution of every agent on all Atari 2600 games is provided in Appendix A.4. Figure best viewed in colour.

**(Q2) Neural plasticity can boost performances of complex deep reinforcement learning approaches, such as Rainbow.** In Fig. 4, we show the learning curves of Rainbow and DQN agents, both with Leaky ReLU baselines, as well as with full and regularised rational plasticity types. While Rainbow is computationally much heavier ($\sim 8$ times slower than DQN in our experiments, with higher memory needs), its rigid form never outperforms the much simpler and more efficient DQN with neural plasticity, and its rational versions dominate in only 1 out of 8 games (Enduro). In our experiments, Rainbow even lost to vanilla DQN on 3 games. These results show that augmenting the plasticity of an RL agent's modeling architecture can be of higher importance than bringing complex and computationally expensive improvements to the learning algorithm.

Therefore, DQN with rational plasticity is a competitive alternative to the complicated and expensive Rainbow method. Plasticity also improves Rainbow agents, answering question **(Q2)** affirmatively.

**(Q3) Neural plasticity directly tackles the overestimation problem.** Revisiting Fig. 4, one can see that Rainbow variants are worst on dynamic environments such as Jamesbond, Time Pilot and particularly Seaquest. For these games, the performance of rigid (Leaky ReLU) DQN progressively decreases. Such drops are well known in the literature and are typically attributed to the overestimation problem of DQN. This overestimation is due to the combination of bootstrapping, off-policy learning and a function approximator (neural network) operating by DQN. van Hasselt et al. showed that inadequate flexibility of the function approximator (either insufficient or excessive) can lead to overestimations of a state-action pairs (2016). The max operator in the update rule of DQN then propagates this overestimation while learning with the replay buffer. The overestimated states can stay in the buffer long before the agent revisit (and thus update) them. This can lead to catastrophic performance drops. To mitigate this problem, van Hasselt et al. introduced a second network to separate action selection from action evaluation, resulting in Double DQN (DDQN).

We have compared rigid DDQN (equipped with Leaky ReLU), to vanilla DQN with neural plasticity on Atari games. As one can see in Fig. 3, DQN with rational plasticity outperforms the more complex (rigid) DDQN algorithm on every considered Atari game. This reinforces the affirmative answer to **(Q1)** from earlier on. More importantly, we have computed the relative overestimation values of the (D)DQN, both with and without neural plasticity, following: overestimation $= \frac{\text{Q-value}-R}{R}$, where the return $R$ corresponds to $R = \sum_{t=0}^{\infty} \gamma^t r_t$, with the observed reward $r_t$ and the discount factor $\gamma$.

The results are summarised in Fig. 5. Plasticity helps to reduce overestimation drastically. DDQN substantially reduces overestimation on Jamesbond, Kangaroo, Tennis, Time Pilot and Seaquest. For these games, DDQN obtains the best performances among all rigid variants only on Jamesbond (*cf.* Tab. 3 in Appendix). Moreover, Fig. 4 reveals that the DDQN agents' performance drops are only delayed and not prevented. The performance drops thus happen after the 200th epoch, after which the agents' training is usually stopped, as no more performance increase seems achievable.

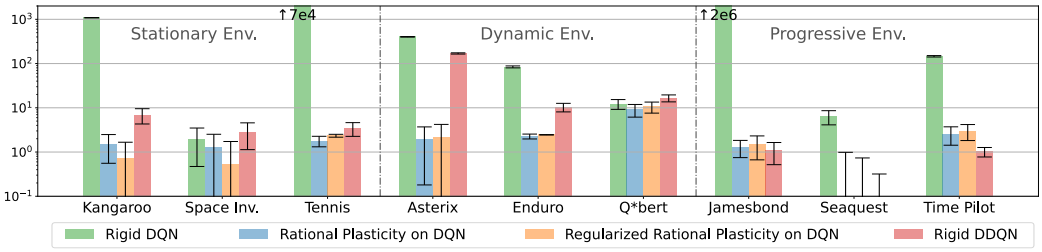



Figure 5: **Plasticity naturally reduces overestimation.** Relative overestimation values (↓, log scale) of rigid DQN and DDQN, as well as DQN with rational and regularised rational plasticity. Each trained agent is evaluated on 100 completed games (5 seeds per game per agent). Agents with rational plasticity lower overestimation values as much or further than rigid DDQN ones, which has specifically been introduced to this end. Figure best viewed in colour.

Overestimation might play a role in the performance drops on progressive environments (*cf.* Fig. 4: Jamesbond, TimePilot and Seaquest), but cannot fully explain the phenomena. RL agents with higher plasticity handle these games much better while embedding only a few more parameters. Hence, we advocate that neural plasticity better deals with distribution shifts of dynamic and progressive environments. Perhaps surprisingly, (regularised) rational plasticity not only works well on challenging progressive environments but also on simpler ones such as Enduro, Pong and Q*bert, where more flexibility is likely to hurt. Flexibility does not lead to overestimation (*cf.* Fig. 5). The rational functions for these games have a simpler profile than ones of more complicated games like Kangaroo and Time Pilot (*cf.* Appendix A.6). The rational functions seem to adapt to the environment's complexity and the policy they need to model. This clearly provides an affirmative answer to **(Q3)**.

**(Q4) Adding parameters through rationals efficiently augments plasticity.** Compared to rigid alternatives, the joint-rational networks embed in total 10 additional parameters and always outperform (*cf.* Fig. 4) PELU (12 more parameters) ones. Our proposed method to add plasticity via rational functions thus efficiently augments the capacity of the network. However, ReLU layers can theoretically approximate rational functions (Telgarsky, 2017). Augmenting the number of layers (or neurons per layer) is thus, theoretically, a costly alternative to augment the plasticity. How many parameters are practically needed in rigid networks to obtain similar performances? Searching for bigger equivalent architectures for RL agents is tedious, as RL training curves possess considerably more variance and noise than SL ones (Pang et al., 2021), but this question is not restricted to RL. We thus answer it by highlighting the generality of our insights, demonstrated by further investigation on a classification scenario (*cf.* Appendix A.1). In short, rigid baselines need up to 3.5 times as many parameters as the architectures that use rational functions in order to obtain similar performances.

All experimental results together clearly show that increasing neural plasticity, particularly through the integration of rational activations functions, considerably benefits deep reinforcement learning agents in a highly computational efficient manner.

## 4 Related Work

Next to the related work discussed throughout the paper, our work on neural plasticity is also related to several research lines on neural architecture search and to the choice of activation functions, particularly in deep reinforcement learning settings.

**The choice of activation functions.** Many functions have been adopted across domains (*e.g.* Leaky ReLU in YOLO (Redmon et al., 2016), hyperbolic tangent in PPO (Schulman et al., 2017), GELU in GPT-3 (Brown et al., 2020)), indicating that the relationship between the choice of activation functions and the performances is highly dependent on the task, architecture and hyper-parameters. As shown, parametric functions augment on plasticity. Molina et al. showed that rational functions can outperform other learnable activation function types on supervised learning tasks (2020). Telgarsky (2017) showed that rationals are locally better approximants than polynomials. Loni et al. (2023) showed that searching for activation functions mitigates the performance drops of sparsity in networks.

**Neural Architectures for Deep Reinforcement Learning.** Cobbe et al. showed that the architecture of IMPALA (Espeholt et al., 2018), notably containing residual blocks, improved the performances over the original Nature-CNN network used in DQN (2019). Motivated by these findings, Pang et al. (2021) recently applied neural architecture search to RL tasks and demonstrated that the optimal architecture highly depend on the environment. Their search provides different architectures across environments, with varying activation functions across layers and potential residual connections. Continuously modifying the complexity of the neural network based on the noisy reward signal in a complex architectural space is extremely resource demanding, particularly for large scale problems. Many reinforcement learning specific problems, such as noisy rewards (Henderson et al., 2018), input interdependency (Mnih et al., 2015), policy instabilities (Haarnoja et al., 2018), sparse rewards, difficult credit assignment (Mesnard et al., 2021), complicate an automated architecture search.

**Plasticity in deep RL.** A lot of attention has recently been brought to the plasticity of RL agents' learning structures. Abbas et al. (2023) have also identified their loss of plasticity and answered it using concatenated ReLU (CReLU) in Rainbow. Nikishin et al. (2022) periodically reset parts of the networks, Sokar et al. (2023) improved the resets by targeting identified dormant neurons. Similarly, Nikishin et al. (2023) inject plasticity via incorporating new trainable weights. Lyle et al. (2022) mitigate capacity (or plasiticity) loss, regularizing some features back to their starting values, and later showed that layer normalization help with plasticity (Lyle et al., 2023). Dohare et al. (2023) tackle dynamics with continual backprop and apply it to RL on PPO (Dohare et al., 2021), also varying between different non-learnable activation functions. Dynamically adapting the hyperparameter landscape is also improves agents' adaptability to distribution shifts (Zahavy et al., 2020; Mohan et al., 2023). Testing how much much of these techniques can be covered by the use of rational plasticity is an interesting line of future work, as rational functions dynamically change the weights optimisation landscape. Fuks et al. (2019) adjust sub-policies on sub-games to find suitable hyperparameters that bootstrap a main evolution-based optimised agent. Apart from using CReLU, all of these techniques are complementary to the use of rational plasticity.

## 5 Limitations, Future Work and Societal Impact

We have shown the benefits of rational activation functions for RL, as a consequence of both their neural and residual plasticity. In our derivation for closure under residuals, we have deduced that the degree of the polynomial in the numerator needs to be greater than that of the denominator. Correspondingly, we have based our empirical investigations on the degrees $(5, 4)$. Interesting future work would be to further automatically select suitable such degrees, or even integrating rationals into dynamic hyperparameters' optimisation techniques. One should also explore neural plasticity in more advanced RL approaches, including short term memory (Kapturowski et al., 2019), neurosymbolic approaches (Delfosse et al., 2024), finer exploration strategy (Badia et al., 2020), and in continual learning (Kudithipudi et al., 2022) techniques. Finally, the noisy optimisation performed in our RL experiments contribute to carbon emissions. However, this is usually a means to an end, as RL algorithms are also used to optimise energy distribution and consumption in several applications.

## 6 Conclusion

In this work, we have highlighted the central role of neural plasticity in deep reinforcement learning algorithms, and have motivated the use of rational activation functions, as a lightweight way of boosting RL agents performances. We derived a condition, under which rational functions embed residual connections. Then the naturally regularised joint-rational activation function was developed, inspired by weight sharing in residual networks.

The simple DQN algorithm equipped with these (regularised) rational forms of plasticity becomes a competitive alternative to more complicated and costly algorithms, such as Double DQN and Rainbow. Fortunately, the complexity of these rational functions also seem to automatically adapt to the one of the environment used for training. Their use could be a substitute for more expensive architectural searches. We thus hope that they will be adopted in future deep reinforcement learning algorithms, as they can provide agents with the necessary neural plasticity required by stationary, dynamic and progressive reinforcement learning environments.

### ACKNOWLEDGEMENTS

The authors thank Elisa Corbean for her help on the manuscript revisions, as well as the anonymous reviewers of ICLR 2024 for their valuable feedback. This research work has been funded by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science within their joint support of the National Research Center for Applied Cybersecurity ATHENE, via the "SenPai: XReLeaS" project, from the German Center for Artificial Intelligence (DFKI). This work was also supported by the project "safeFBDC - Financial Big Data Cluster (FKZ: 01MK21002K)", funded by the German Federal Ministry for Economics Affairs and Energy as part of the GAIA-x initiative. It benefited from the Hessian Ministry of Higher Education, Research, Science and the Arts (HMWK; project "The Third Wave of AI")

## REFERENCES

Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C. Machado. Loss of plasticity in continual deep reinforcement learning. *ArXiv*, 2023.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies. In *8th International Conference on Learning Representations (ICLR)*, 2020.

Chayan Banerjee, Zhiyong Chen, and Nasimul Noman. Improved soft actor-critic: Mixing prioritized off-policy samples with on-policy experience. *ArXiv*, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *ArXiv*, 2017.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Karl Cobbe, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning, (ICML)*, 2019.

Quentin Delfosse, Hikaru Shindo, Devendra Singh Dhami, and Kristian Kersting. Interpretable and explainable logical policies via neurally guided symbolic abstraction. 2023a.

Quentin Delfosse, Wolfgang Stammer, Thomas Rothenbacher, Dwarak Vittal, and Kristian Kersting. Boosting object representation learning via motion and object continuity. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML)*, 2023b.

Quentin Delfosse, Sebastian Sztwiertnia, Wolfgang Stammer, Mark Rothermel, and Kristian Kersting. Interpretable concept bottlenecks to align reinforcement learning agents. *ArXiv*, 2024.

Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Mushroomrl: Simplifying reinforcement learning research. *ArXiv*, 2020.

Shibhansh Dohare, Ashique Rupam Mahmood, and Richard S. Sutton. Continual backprop: Stochastic gradient descent with persistent randomness. *ArXiv*, 2021.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Parash Rahman, Richard S. Sutton, and A. Rupam Mahmood. Maintaining plasticity in deep continual learning. *ArXiv*, 2023.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 2018.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

Jesse Farebrother, Marlos C. Machado, and Michael Bowling. Generalization and regularization in DQN. *ArXiv*, 2018.

Lior Fuks, Noor H. Awad, Frank Hutter, and Marius Thomas Lindauer. An evolution strategy with progressive episode lengths for playing games. In *International Joint Conference on Artificial Intelligence*, 2019.

D Garlick. Understanding the nature of the general factor of intelligence: the role of individual differences in neural plasticity as an explanatory mechanism. *Psychological review*, 2002.

Mariana-Iuliana Georgescu, Radu Tudor Ionescu, Nicolae-Catalin Ristea, and Nicu Sebe. Non-linear neurons with human-like apical dendrite activations. *ArXiv*, 2020.

Albert Gidon, Timothy Adam Zolnik, Pawel Fidzinski, Felix Bolduan, Athanasia Papoutsi, Panayiota Poirazi, Martin Holtkamp, Imre Vida, and Matthew Evan Larkum. Dendritic action potentials and computation in human layer 2/3 cortical neurons. *Science*, 2020.

Luke B. Godfrey. An evaluation of parametric activation functions for deep learning. *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019.

Mohit Goyal, Rajan Goyal, and Brejesh Lall. Learning activation functions: A new paradigm of understanding neural networks. *ArXiv*, 2019.

Klaus Greff, Rupesh Kumar Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. In *5th International Conference on Learning Representations (ICLR)*, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *9th International Conference on Learning Representations (ICLR)*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second Conference on Artificial Intelligence AAAI*, 2018.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the Thirty-Second Conference on Artificial Intelligence (AAAI)*, 2018.

Steven Kapturowski, Georg Ostrovski, John Quan, Rémi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *7th International Conference on Learning Representations (ICLR)*, 2019.

Srivatsan Krishnan, Amir Yazdanbaksh, Shvetank Prakash, Jason J. Jabbour, Ikechukwu Uchendu, Susobhan Ghosh, Behzad Boroujerdian, Daniel Richins, Devashree Tripathy, Aleksandra Faust, and Vijay Janapa Reddi. Archgym: An open-source gymnasium for machine learning assisted architecture design. *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Dhireesha Kudithipudi, Mario Aguilar-Simon, Jonathan Babb, Maxim Bazhenov, Douglas Blackiston, Josh C. Bongard, Andrew P. Brna, Suraj Chakravarthi Raja, Nick Cheney, Jeff Clune, Anurag Reddy Daram, Stefano Fusi, Peter Helfer, Leslie M. Kay, Nicholas A. Ketz, Zsolt Kira, Soheil Kolouri, Jeffrey L. Krichmar, Sam Kriegman, Michael Levin, Sandeep Madireddy, Santosh Manicka, Ali Marjaninejad, Bruce L. McNaughton, Risto Miikkulainen, Zaneta Navratilova, Tej Pandit, Alice Parker, Praveen K. Pilly, Sebastian Risi, Terrence J. Sejnowski, Andrea Soltoggio, Nicholas Soures, Andreas Savas Tolias, Darío Urbina-Meléndez, Francisco J. Valero-Cuevas, Gido M. van de Ven, Joshua T. Vogelstein, Felix Wang, Ron Weiss, Angel Yanguas-Gil, Xinyun Zou, and Hava T. Siegelmann. Biological underpinnings for lifelong learning machines. *Nature Machine Intelligence*, 2022.

Qianli Liao and Tomaso A. Poggio. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *ArXiv*, 2016.

Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. SPACE: unsupervised object-oriented scene representation via spatial attention and decomposition. In *8th International Conference on Learning Representations (ICLR)*, 2020.

Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *15th European Conference on Computer Vision (ECCV)*, 2018.

Mohammad Loni, Aditya Mohan, Mehdi Asadi, and Marius Thomas Lindauer. Learning activation functions for sparse neural networks. *ArXiv*, 2023.

Liang Lu and Steve Renals. Small-footprint deep neural networks with highway connections for speech recognition. In *17th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2016.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. *ArXiv*, 2022.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Ávila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. *ArXiv*, 2023.

Franco Manessi and Alessandro Rozza. Learning combinations of activation functions. In *24th International Conference on Pattern Recognition (ICPR)*, 2018.

Thomas Mesnard, Theophane Weber, Fabio Viola, Shantanu Thakoor, Alaa Saade, Anna Harutyunyan, Will Dabney, Thomas S. Stepleton, Nicolas Heess, Arthur Guez, Eric Moulines, Marcus Hutter, Lars Buesing, and Rémi Munos. Counterfactual credit assignment in model-free reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

Diganta Misra. Mish: A self regularized non-monotonic activation function. In *31st British Machine Vision Conference 2020, (BMVC)*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas 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*, 2015.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

Aditya Mohan, C. E. Benjamins, Konrad Wienecke, Alexander Dockhorn, and Marius Lindauer. Autorl hyperparameter landscapes. *ArXiv*, 2023.

Alejandro Molina, Patrick Schramowski, and Kristian Kersting. Padé activation units: End-to-end learning of flexible activation functions in deep networks. In *8th International Conference on Learning Representations (ICLR)*, 2020.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron C. Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning*, 2022.

Evgenii Nikishin, Junhyuk Oh, Georg Ostrovski, Clare Lyle, Razvan Pascanu, Will Dabney, and André Barreto. Deep reinforcement learning with plasticity injection. *ArXiv*, 2023.

Johan S. Obando-Ceron and Pablo Samuel Castro. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

Dong Pang, Xinyi Le, and Xinping Guan. RL-DARTS: differentiable neural architecture search via reinforcement-learning-based meta-optimizer. *Knowl. Based Syst.*, 2021.

Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Julien Roy, Paul Barde, Félix G. Harvey, Derek Nowrouzezahrai, and Chris Pal. Promoting coordination through policy regularization in multi-agent deep reinforcement learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020 (NeurIPS)*, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv*, 2017.

Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *International Conference on Machine Learning*, 2016.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning*, 2023.

Matus Telgarsky. Neural networks and rational functions. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, 2017.

Ludovic Trottier, Philippe Giguère, and Brahim Chaib-draa. Parametric exponential linear unit for deep convolutional neural networks. In *IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2017.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth Conference on Artificial Intelligence (AAAI)*, 2016.

Andreas Veit, Michael J. Wilber, and Serge J. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

Yue Wu, Yewen Fan, Paul Pu Liang, Amos Azaria, Yuanzhi Li, and Tom M Mitchell. Read and reap the rewards: Learning to play atari with the help of instruction manuals. *Advances in Neural Information Processing Systems*, 2024.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *7th International Conference on Learning Representations (ICLR)*, 2019.

Changqing Xu, Mingyue Wang, and Xian Li. Generalized vandermonde tensors. *Frontiers of Mathematics in China*, 2016.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *9th International Conference on Learning Representations, (ICLR)*, 2021.

Tom Zahavy, Zhongwen Xu, Vivek Veeriah, Matteo Hessel, Junhyuk Oh, H. V. Hasselt, David Silver, and Satinder Singh. A self-tuning actor-critic algorithm. *arXiv: Machine Learning*, 2020.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations (ICLR)*, 2017.

# A APPENDIX

As mentioned in the main body, the appendix contains additional materials and supporting information for the following aspects: rational activation functions improving plasticity (4), comparison of rational and rigid networks with different sizes on supervised learning experiments (A.1), results on replacing residual blocks with rational activation functions (A.2), every final and maximal scores obtained by the reinforcement learning agents used in our experiments (A.3), the evolutions of these scores (A.4), the different environment types with illustrations of their changes (A.5), graphs of the learned rational activation functions (A.6) and technical details for reproducibility (A.8).

Rational functions improve plasticity

To prove that rational can help with plasiticy, we tested them in continual learning settings (with more abrupt distribution shifts). We included Concatenated RELU and rational functions to an existing implementation of continual AI[3], in which 4 layers (2 convolutional ones and to fully connected ones) networks are trained on MNIST. The network then continues training on PERM.1, a variation of the dataset, for which a fixed random permutation is applied to every image. Another permutation is used for PERM. 2, used after the training on PERM1. As shown in Fig. 6, networks with rationals are both better at modelling the new data (higher accuracies on the currently trained data), but are also able to retain more information about the data previously trained on. Networks with Continual ReLU (Shang et al., 2016) better retain information on Task 1, while performing on par with ReLU ones for the 2 other tasks.

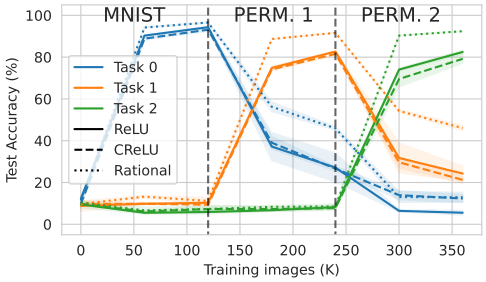



Figure 6: **Rational function improve plasticity on the permutted MNIST experiment.** Rational networks obtain better accuracies on each currently and previously trained datasets.

## A.1 RATIONAL EFFICIENT PLASTICITY CAN REPLACE LAYER'S WEIGHT PLASTICITY

We here show that networks with rational activations not only outperform Leaky ReLU ones with the same amount of parameters, but also to outperform deeper and more heavily parametrised neural networks (indicated by the colours). For example, a rational activated VGG4 not only performs better than a rigid Leaky ReLU VGG4 at 1.37M parameters, but even performs similarly to the 4.71M parameters rigid VGG6. Activation's plasticity allowing to reduce the number of layers weights is also shown by the experiments summarized in Tab. 2 in the next section, where blocks from a pretrained ResNet are replaced by a rational function, and the resulting networks are able to recover and surpass their accuracies.

| Architecture | | VGG4 | | VGG6 | | VGG8 | |
|---|---|---|---|---|---|---|---|
| Activation function | | LReLU | Rational | LReLU | Rational | LReLU | Rational |
| CIFAR 10 | Training Acc@1 | 83.0±.3 | 87.1±.6 | 86.9±.2 | 89.2±.2 | 90.1±.1 | 92.4±.2 |
| | Testing Acc@1 | 80.0±1. | 84.3±.5 | 83.1±.6 | 85.4±.6 | 85.0±1. | 86.9±.3 |
| CIFAR 100 | Training Acc@1 | 64.6±.8 | 70.4±.9 | 70.7±.6 | 86.0±.9 | 87.7±.2 | 87.8±.1 |
| | Testing Acc@1 | 56.5±.9 | 58.9±.6 | 59.0±.5 | 59.9±.9 | 60.0±.9 | 59.9±.4 |
| # Network parameters | | 1.37M | | 4.71M | | 9.27M | |

Table 1: Shallow rational networks perform as deeper Leaky ReLU ones. VGG networks training and testing top-1 accuracies with different numbers of layers are evaluated on CIFAR10 and CIFAR100. Rational VGG4 has similar performances as VGG6 network, with 3.5 times less parameters, and Rational VGG6 outperforms VGG8, with two times less parameters. Shaded colour pairs included for emphasis.

[3] https://github.com/ContinualAI/colab/blob/master/notebooks/permuted_and_split_mnist.ipynb

### A.2 RESIDUAL BLOCK LEARN OF DEEP RESNET LEARN ACTIVATION FUNCTION-LIKE BEHAVIOUR.

We present in this section lesioning experiments, where a residual block is lesioned from a pretrained Residual Network, and the surrounding blocks are fine-tuned (with a learning rate of $0.001$) for $15$ epochs. These lesioning experiments were first conducted by Veit et al. (2016). We also perform rational lesioning, where we replace a block by an (identity initialised)[4] rational activation function (instead of removing the block), and train the activation function along with the surrounding blocks. The used rational functions have the same order as in every other experiment ($(m, n) = (5, 4)$), that satisfies the rational residual property derive in the paper). We report recovery percentages, computed following:

$$\text{recovery} = 100 \times \frac{\text{finetuned} - \text{surgered}}{\text{original} - \text{surgered}}. \tag{1}$$

We also provide the amount of dropped parameters of each lesioning.

Table 2: Rational functions improve lesioning. The recovery percentages for finetuned networks after lesioning (Veit et al., 2016) of a ResNet layer's (L) block (B) are shown. Residual blocks were lesioned, *i.e.* replaced with the identity (Base) or a rational from a pretrained ResNet101 (44M parameters). Then, the surrounding blocks (and implanted rational activation function) are retrained for 15 epochs. Larger percentages are better, best results are in **bold**.

| Recovery (%) | Lesioning | L2B3 | L3B19 | L3B22 | L4B2 |
|---|---|---|---|---|---|
| Training | Original (Veit et al., 2016) | 100.9 | 90.5 | 100 | 58.9 |
| | Rational (ours) | **101.1** | **104** | **120** | **91.1** |
| Testing | Original (Veit et al., 2016) | **93.1** | 97.1 | 81.6 | 81.7 |
| | Rational (ours) | 90.5 | **97.6** | **91.5** | **85.3** |
| % dropped params | | 0.63 | 2.51 | 2.51 | 10.0 |

As the goal is to show that flexible rational functions can achieve similar modelling capacities to the residual blocks, we did not apply regularisation methods and mainly focused on training accuracies. We can clearly observe that rational activation functions lead to performance improvements that even surpass the original model, or are able to maintain performances when the amount of dropped parameters rises.

### A.3 COMPLETE SCORES TABLE FOR DEEP REINFORCEMENT LEARNING

Through this work, we showed the performance superiority of reinforcement learning agents that embed additional plasticity provided by learnable rational activation functions. We used human normalised scores (*cf.* Eq. 2) for readability. For completeness, we provide in this section the final raw scores of every trained agent. As many papers provide the maximum obtained score among every epoch and every agent, even if we consider it to be an inaccurate and noisy indicator of the performances, for which random actions can still be taken (because of $\epsilon$-greedy strategy also being used in evaluation). A fairer indicator to compare methods is the mean score. We thus also provide final mean scores (of agents retrained among 5 seeded reruns) with standard deviation. We start off by providing the human scores used for normalisation (provided by van Hasselt et al., in Table 5), then provide final mean and maximum obtained raw scores of every agent.

[4] all weights are initially set to 0 but $a_1$ (and $b_0$), both set to 1.

| Algorithm | DQN | | | DDQN | DQN with Plasticity | | |
|---|---|---|---|---|---|---|---|
| Activation | LReLU | SiLU | d+SiLU | LReLU | PELU | rational | joint-rational |
| Asterix | 1.85±1.2 | 0.52±0.6 | 2.14±1.4 | 48.9±17.7 | 25.8±3.7 | **242**±23.5 | 168±32.6● |
| Battlezone | 11.4±7.0 | 21.2±15.0 | 11.3±6.7 | 68.2±34.8 | 46.6±19.5 | 70.1±2.1● | **77.4**±8.7 |
| Breakout | 558±166 | 93.9±57.6 | 11.7±14.0 | 286±122 | 788±79.2 | 1134±130● | **1210**±36.0 |
| Enduro | 16.3±21.3 | 37.0±17.7 | 0.37±0.5 | 47.7±18.1 | 24.5±42.6 | **141**±15.0 | 129±14.7● |
| Jamesbond | 8.62±6.4 | 6.08±3.7 | 5.28±4.4 | 10.7±11.1 | 74.2±51.5 | 308±48.5● | **312**±59.5 |
| Kangaroo | 11.8±12.5 | 128±95.6● | 13.9±18.5 | 17.2±14.5 | 57.7±14.6 | 107±43.1 | **193**±86.8 |
| Pong | 101±5.5 | 96.1±12.0 | 104±3.3 | 91.3±30.8 | 106.4±2.2 | 107.0±2.4● | **107.3**±2.7 |
| Qbert | 55.4±17.1 | 14.2±17.0 | 2.74±0.2 | 74.0±21.7 | 101±6.6 | **120**±2.8 | 117±4.9● |
| Seaquest | 0.57±0.4 | 3.67±4.1 | 0.18±0.2 | 2.17±0.9 | 9.21±2.5 | 16.3±0.5● | **18.4**±3.3 |
| Skiing | -90.7±37.9 | -111±-0.7 | -85.5±43.4 | -86.9±46.6 | -111±-.7 | **-59.5**±60.7 | -60.2±56.1● |
| Space Inv. | 33.9±4.3 | 33.1±11.9 | 32.4±12.4 | 31.0±1.0 | 50.1±3.3● | 42.3±3.1 | **95.1**±17.7 |
| Tennis | 8.94±17.3 | 26.3±53.3 | 78.5±64.3 | 32.1±51.6 | 106±53.3 | 257.8±2.8● | **258.3**±5.2 |
| Timepilot | 14.9±14.3 | 19.3±31.0 | 18.3±38.1 | 6.61±7.5 | 124±26.1 | **341**±105 | 253±11.0● |
| Tutankham | 0.03±2.8 | 58.2±48.6 | 2.89±4.0 | 24.4±-0.4 | 91.6±29.3 | 130±10.7● | **134**±29.3 |
| Videopinball | 440±123 | 55.8±61.9 | -4.03±32.5 | 626±241 | 299±168 | **1616**±1026 | 906±539● |
| **# Wins** | 0/15 | 0/15 | 0/15 | 0/15 | 0/15 | 6/15 | 9/15 |
| **# Super-Human** | 3/15 | 1/15 | 1/15 | 2/15 | 6/15 | **11/15** | **11/15** |

Table 3: Neural plasticity leads to vast performance improvements. Normalised mean scores and standard deviations (in percentage, *cf.* Appendix A.8 for the equation) of rigid baselines (*i.e.* DQN and DDQN with Leaky ReLU, DQN with SiLU and SiLU + dSiLU), as well as DQN with plasticity: using PELU, rational (full) and joint-rational (regularised), are reported over five experimental random seeded repetitions (larger mean values are better). The best results are highlighted in **bold** and runner-ups denoted with • markers. The last rows summarise the number of times best mean scores were obtained by each agent and the number of super-human performances.

**Final mean and maximum obtained scores of Rainbow agents:**

| Evaluation | **Final Mean Scores** | | | Max. Obtained Scores | | |
|---|---|---|---|---|---|---|
| Plasticity | rigid | full | regularised | rigid | full | regularised |
| Breakout | 52 | 279 | **303** | 383 | **569** | **569** |
| Enduro | 844 | **1473** | 1470 | 1388 | **1973** | 1964 |
| Kangaroo | 40 | **2157** | 2139 | **6300** | 6000 | 4800 |
| Q*bert | 149 | **11931** | 11551 | 16125 | **23550** | **23550** |
| Seaquest | 82 | 247 | **282** | 920 | **1280** | **1280** |
| Space Inv. | 595 | **1263** | 1157 | 2070 | **3395** | 2875 |
| Time Pilot | 3926 | 5386 | **6411** | 12700 | **15900** | **15900** |

Table 4: Final mean and maximum obtained scores obtained by rigid Rainbow agents (*i.e.* using Leaky ReLU), as well as Rainbow with full (*i.e.* using rational activation functions) and regularised (*i.e.* using joint-rational ones) plasticity (only 1 run because of computational cost, larger values are better).

**Final mean scores of all agents:**

| Algorithm | Random | DQN | | | DDQN | DQN with Plasticity | | |
|---|---|---|---|---|---|---|---|---|
| Network type | - | LReLU | SiLU | d+SiLU | LReLU | PELU | full | regularised |
| Asterix | 67.9±2.2 | 206±90 | 107±45 | 228±108 | 3723±1324 | 1998±275 | **18109±1755** | 12621±2436 |
| Battlezone | 788±38 | 4464±2291 | 7612±4877 | 4429±2183 | 22775±11265 | 15807±6320 | 23403±701 | **25749±2837** |
| Breakout | 0.14±01 | 155±46 | 26.2±16 | 3.4±3.89 | 79.4±33.8 | 219±22 | 315±36 | **336±10** |
| Enduro | 0±0 | 121±158 | 274±131 | 2.77±3.41 | 353±134 | 181±315 | **1043±111** | 957±109 |
| Jamesbond | 6.39±0.41 | 37.6±23.6 | 28.4±13.8 | 25.5±16.2 | 45.2±40.7 | 275±187 | 1122±176 | **1137±216** |
| Kangaroo | 14.2±0.9 | 335±342 | 3500±2607 | 393±504 | 484±395 | 1586±398 | 2940±1175 | **5266±2365** |
| Pong | -20.2±0 | 15.9±2 | 14.1±4.3 | 16.9±1.2 | 12.4±11 | 17.8±0.8 | 18±0.9 | **18.1±1** |
| Q*bert | 40.6±2.8 | 6715±2058 | 1754±2048 | 371±28 | 8954±2616 | 12143±795 | **14436±336** | 14080±593 |
| Seaquest | 20.1±0.4 | 250±162 | 1504±1677 | 94.6±87.2 | 898±353 | 3740±991 | 6603±200 | **7461±1321** |
| Skiing | -16104±92 | -27365±4794 | -29890±4 | -26725±5485 | -26892±5881 | -29912±10 | **-23487±7624** | -23582±7058 |
| Space Inv. | 51.6±1.1 | 531±62 | 520±169 | 509±176 | 490±15 | 759±48 | 650±45 | **1395±251** |
| Tennis | -23.9±0.0 | -22.4±3.0 | -19.4±9.2 | -10.4±11.1 | -18.4±8.9 | -5.6±9.2 | 20.5±0.5 | **20.6±0.9** |
| TimePilot | 688±30 | 1428±739 | 1644±1566 | 1594±1918 | 1016±401 | 6818±1323 | **17632±5242** | 13261±576 |
| Tutankham | 3.51±0.54 | 3.55±4.3 | 81.9±66 | 7.41±5.96 | 36.4±0 | 127±40 | 179±15 | **184±40** |
| VideoPinb. | 6795±461 | 45683±11383 | 11730±5941 | 6439±3336 | 62151±21791 | 42051±15356 | **149712±91219** | 86942±48143 |

Table 5: Final mean raw scores (with std. dev.) of rigid baselines (*i.e.* DQN and DDQN with Leaky ReLU, DQN with SiLU and SiLU + dSiLU), as well as DQN with full plasticity (*i.e.* using rational activation functions) and regularised plasticity (*i.e.* using joint-rational ones) on Atari 2600 games, averaged over 5 seeded reruns (larger mean values are better).

**Maximum obtained scores:**

| Algorithm | Random | DQN | | | DDQN | DQN with Plasticity | | |
|---|---|---|---|---|---|---|---|---|
| Network type | - | LReLU | SiLU | d+SiLU | LReLU | PELU | full | regularised |
| Asterix | 71 | 9250 | 3400 | 3800 | 20150 | 9300 | 84950 | 49700 |
| Battlezone | 843 | 88000 | 81000 | 70000 | 97000 | 68000 | 78000 | 94000 |
| Breakout | 0 | 427 | 370 | 344 | 411 | 430 | 864 | 864 |
| Enduro | 0 | 1243 | 928 | 1041 | 1067 | 1699 | 1946 | 1927 |
| Jamesbond | 6 | 5600 | 5750 | 700 | 7500 | 6150 | 9250 | 13300 |
| Kangaroo | 15 | 14800 | 15600 | 10200 | 13000 | 12400 | 16200 | 16800 |
| Pong | -20 | 21 | 21 | 21 | 21 | 21 | 21 | 21 |
| Q*bert | 45 | 19425 | 11700 | 5625 | 19200 | 18900 | 24325 | 25075 |
| Seaquest | 20 | 7440 | 8300 | 740 | 15830 | 14860 | 9100 | 26990 |
| Skiing | -15997 | -5987 | -6505 | -6267 | -5359 | -5495 | -5368 | -5612 |
| Space Inv. | 53 | 2435 | 2205 | 2460 | 2290 | 2030 | 2490 | 3790 |
| Tennis | -23 | 8 | 1 | -1 | 4 | -1 | 24 | 36 |
| Time Pilot | 730 | 11900 | 15500 | 12500 | 12200 | 16300 | 72000 | 28000 |
| Tutankham | 4 | 249 | 267 | 267 | 274 | 397 | 334 | 309 |
| VideoPinb. | 7599 | 998535 | 950250 | 338512 | 991669 | 322655 | 997952 | 998324 |

Table 6: Maximum obtained scores (with std. dev.) of rigid baselines (*i.e.* DQN and DDQN with Leaky ReLU, DQN with SiLU and SiLU + dSiLU), as well as DQN with full plasticity (*i.e.* using rational activation functions) and regularised plasticity (*i.e.* using joint-rational ones) on Atari 2600 games, averaged over 5 seeded reruns (larger values are better).

**Human scores used for normalisation:**
Asterix: 7536, Battlezone: 33030, Breakout: 27.9, Enduro: 740.2, Jamesbond: 368.5, Kangaroo: 2739, Pong: 15.5, Q*bert: 12085, Seaquest: 40425.8, Skiing: $-3686.6$, Space Invaders: 1464.9, Tennis: $-6.7$, Time Pilot: 5650, Tutankham: 138.3, Video Pinball: 15641.1

### A.4 EVOLUTION OF THE SCORES ON EVERY GAME

The main part present some graphs that compares performance evolutions of the Rainbow and DQN agents with plasticity, as well as Rigid DQN, DDQN and Rainbow agents. We here provide the evolution of the scores of every tested DQN and the DDQN agents on the complete game set. DQN agents with higher plasticity are always the best-performing ones. Experiments on several games (e.g. Jamesbond, Seaquest) show that using DDQN does not prevent the performance drop but only delays it.


Figure 7: Smoothed (*cf.* Eq. 3) evolutions of the scores on every tested game for DQN agents with full (*i.e.* using rational activation functions) and regularised (*i.e.* using joint-rational ones) plasticity, and original DQN agents using Leaky ReLU, SiLU and SiLU+dSiLU, as well as for DDQN agents with Leaky ReLU.

### A.5 ENVIRONMENTS TYPES: STATIONARY, DYNAMICS AND PROGRESSIVE

The used environments have been separated in 3 categories, describing their potential changes through agents learning. This categorisation is here illustrated with frames of the tested games. As one can see: Breakout, Kangaroo, Pong, Skiing, Space Invaders, Tennis, Tutankham and VideoPinball can be categorised as **stationary environment**, as changes are minimal for the agents in these games. Asterix, BattleZone, Q*bert and Enduro present environment changes, that are early reached by the playing agents, and are thus **dynamic environments**. Finally, Jamesbond, Seaquest and Time Pilot correspond to **progressive environments**, as the agents needs to master early changes to access new parts of these environments.

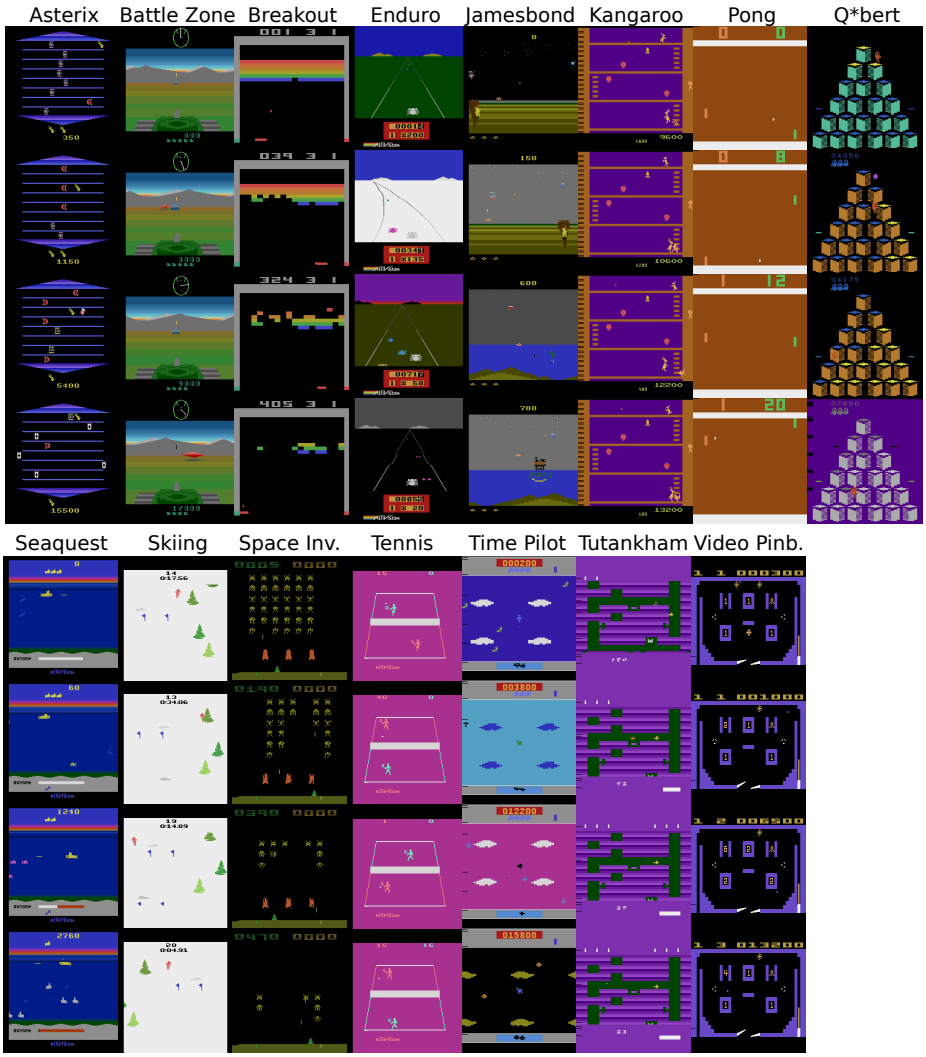
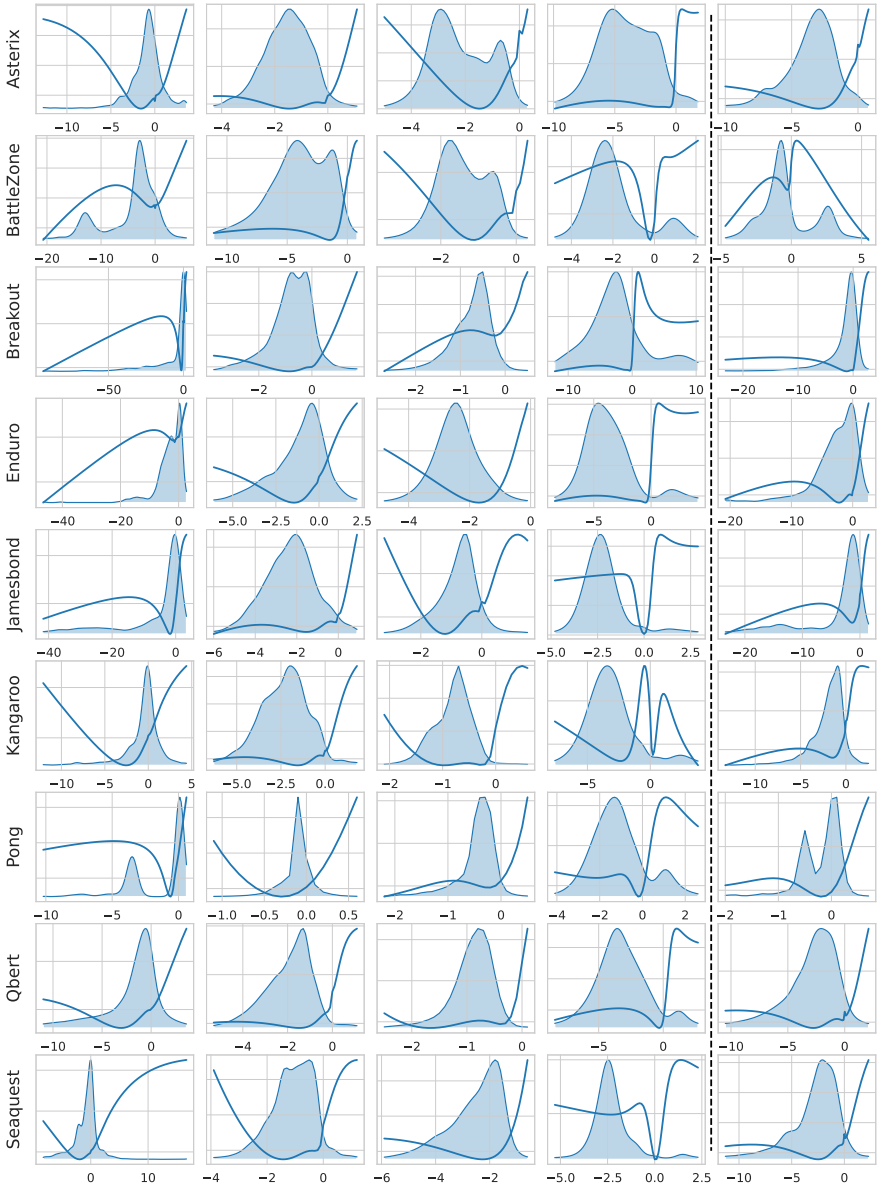



Figure 8: Images extracted from DQN agents with full plasticity playing the set of 15 Atari 2600 games used in this paper. Stationary environments (e.g. Pong, Video Pinball) do not evolve during training, dynamic ones provide different input/output distributions that are early accessible in the game (e.g Q*bert, Enduro) and progressive ones (e.g. Jamesbond, Time Pilot) require the agent to improve for the it to evolve.

### A.6 LEARNED RATIONAL ACTIVATION FUNCTIONS

We have explained in the main text how rational functions of agents used on different games can exhibit different complexities. This section provides the learned parametric rational functions learned by DQN agents with full plasticity (left) and by those with regularised plasticity (right) after convergence for every different tested game of the gym Atari 2600 environment. Kernel Density Estimations (with Gaussian kernels) of input distributions indicates where the functions are most activated. Rational functions from agents trained on simpler games (*e.g.* Enduro, Pong, Q*bert) have simpler profiles (*i.e.* fewer distinct extremas).



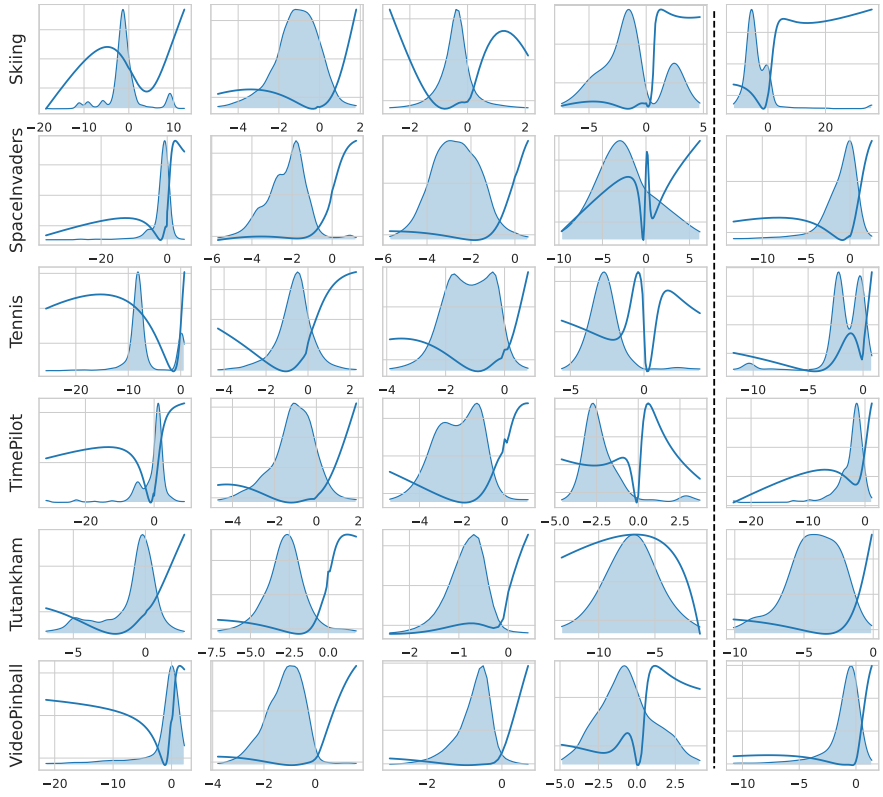



Figure 9: Profiles (dark blue) and input distributions (light blue) of rational functions (left) and joint-rational ones (right) of DQN agents on the different tested games. (Joint-)rational functions from agents of simpler games have simpler profiles (*i.e.* fewer distinct extrema).

### A.7 Evolution of rationals on the Perm-MNIST continual learning experiment

Figure 10 depicts the evolutions of rational functions through the permuted MNIST experiment. One can see that while the function of the first layer remains stable through the successive datasets, the second one flatten at its most activated region (around 0), while the third one increase its slope in this region, leading to higher gradients. This suggests that rational functions can help adapting the gradient scales at each layer. Further investigating this is an interesting line of future work.

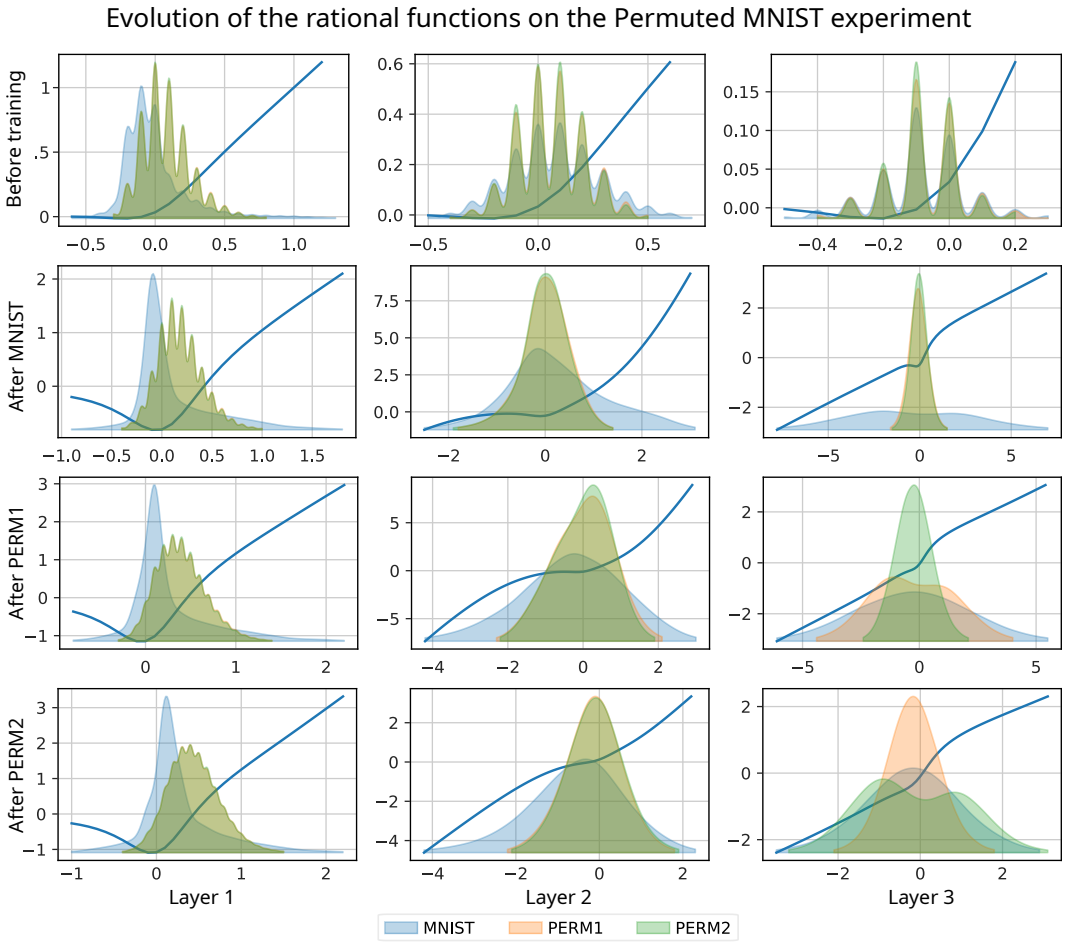



Figure 10: Evolution of the rational activation functions on the permuted MNIST experiment (*cf.* 4). The 3 rational activation functions used for training (and retraining) are adapting to fit the data (depicted in semi transparent).

### A.8 Technical details to reproduce the experiments

We here provide details on our experiments for reproducibility. **We used the seed 0, 1, 2, 3, 4 for every multi-seed experiment.**

#### Supervised Learning Experiments

For the lesioning experiment, we used an available[5] pretrained Residual Network. We then remove the corresponding block (and potentially replace it with an identity initialised rational activation function) (surgered). We finetune the new models, allowing for optimisation of the previous and next layers (and potentially the rational function) for 15 epochs with SGD (learning rate of 0.001).

[5] https://download.pytorch.org/models/resnet101-5d3b4d8f.pth

For the classification experiments, we run on CIFAR10 and CIFAR100 (Krizhevsky et al., MIT License), we let every network learn for 60 epochs. We use the code provided by Molina et al. (2020), with only one classification layer in these smaller VGG versions (VGG4, VGG6 and VGG8, against 3 for VGG16 and larger). We use SGD as the optimisation algorithm, with a learning rate of $0.02$ and $128$ as batch size. The VGG networks contain successive VGG blocks that all consist of $n$ convolutional layers, $i$ input channels and $o$ output channels, stride 3 and padding 1, followed by an activation function, and 1 Max Pooling layer. For each used architecture, the ($n$, $i$, $o$) parameters of the successive blocks are:

- VGG4: $(1, 3, 64) \rightarrow (1, 64, 128) \rightarrow (2, 128, 256)$
- VGG6: $(1, 3, 64) \rightarrow (1, 64, 128) \rightarrow (2, 128, 256) \rightarrow (2, 256, 512)$
- VGG8: $(1, 3, 64) \rightarrow (1, 64, 128) \rightarrow (2, 128, 256) \rightarrow (2, 256, 512) \rightarrow (2, 512, 512)$

The output of these blocks is then passed on to a classifier (linear layer). Only activation functions differ between the Leaky ReLU and the Rational versions.

### Reinforcement Learning Experiments

To ease the reproducibility of our the reinforcement learning experiments, we used the *Mushroom RL* library (D'Eramo et al., 2020) on the Arcade Learning Environment (GNU General Public License). We used states consisting of 4 consecutive grey-scaled images, downsampled to $84 \times 84$. Computing the gradients for rational functions takes longer than e.g. ReLU. However, we used a CUDA optimized implementation of the rational activation functions that we open source along with this paper. In practice, we did not notice any significant training time difference.

**Network Architecture.** The input to the network is thus a 84x84x4 tensor containing a rescaled, and gray-scaled, version of the last four frames. The first convolution layer convolves the input with 32 filters of size 8 (stride 4), the second layer has 64 layers of size 4 (stride 2), the final convolution layer has 64 filters of size 3 (stride 1). This is followed by a fully-connected hidden layer of 512 units. All these layers are separated by the corresponding activation functions (either Leaky ReLU, SiLU, SiLU for convolution layers and dSiLU for linear ones, PELU, rational functions (at each layer) and joint-rational ones (shared accross layers) of order $m = 5$ and $n = 4$, initialised to approximate Leaky ReLU). We used the default PeLU initial hyperparameters (a=1, b=1, c=1) and let the weights optimizer tune them through training, as for rational functions. For CRELU, we took the implementation from ML Compiled [6], and halves the number of filters in the following convolutional layers to keep the same network structure intact, as done by Shang et al. (2016).

**Hyper-parameters.** We evaluate the agents every $250$K steps, for $125$K steps. The target network is updated every $10$K steps, with a replay buffer memory of initial size $50$K, and maximum size $500$K, except for Pong, for which all these values are divided by $10$. The discount factor $\gamma$ is set to $0.99$ and the learning rate is $0.00025$. We do not select the best policy among seeds between epochs. We use the simple $\epsilon$-greedy exploration policy, with the $\epsilon$ decreasing linearly from $1$ to $0.1$ over 1M steps, and an $\epsilon$ of $0.05$ is used for testing.

The only difference from the evaluation of Mnih et al. (2015) and of van Hasselt et al. (2016) evaluation is the use of the Adam optimiser instead of RMSProp, for every evaluated agent.

**Normalisation techniques.** To compute human normalised scores, we used the following equation:

$$\text{score}_{\text{normalised}} = 100 \times \frac{\text{score}_{\text{ agent}} - \text{score}_{\text{ random}}}{\text{score}_{\text{ human}} - \text{score}_{\text{ random}}}, \tag{2}$$

For readability, the curves plotted in the Fig. 4 and Fig. 8 are smoothed following:

$$score_t = \alpha \times score_{t-1} + (1 - \alpha) \times scores_t, \tag{3}$$

with $\alpha = 0.9$.

**Overestimation computation**. We used the following formulae to compute relative overestimation.

$$\text{overestimation} = \frac{\text{Q-value} - R}{R} \tag{4}$$

[6] https://ml-compiled.readthedocs.io/en/latest/activations.html

### RL NETWORK ARCHITECTURE

The DQN, DDQN and Rainbow agents networks architecture, rational plasticity (with rational activations functions at each layer) and of the regularized ones (with one joint-rational activation function shared across layers). For the other activation functions, the "Rat." blocks are replaced with Leaky ReLU, CReLU, SiLU, or PELU. For the d+SiLU networks, SiLU is used on the convolutional layers (*i.e.* first two), and dSiLU in the fully connected ones (*i.e.* last two).

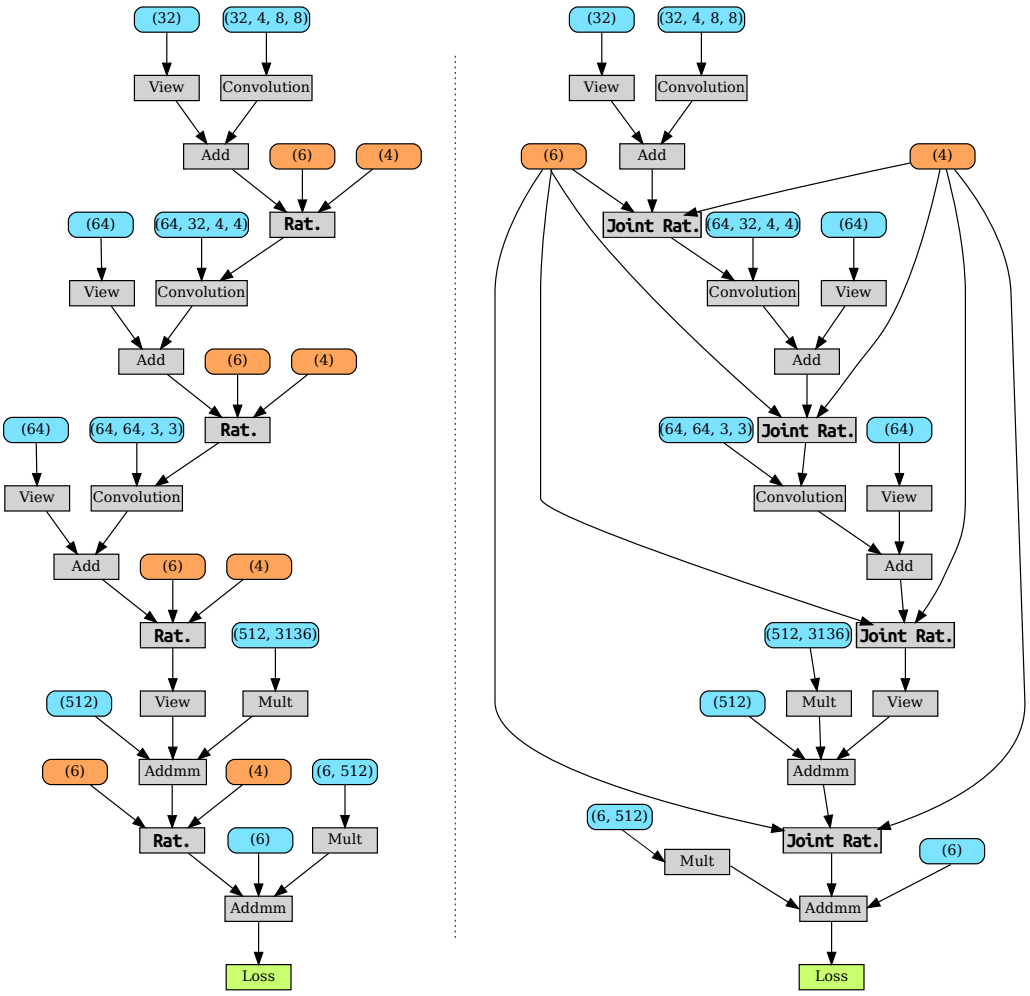



Figure 11: left: The DQN agents' neural network equipped with Rational Activation Functions (Rat.). Any other network with classical activation functions (as Leaky Relu or SiLU) would be similar, with the corresponding activation function instead of the rational one. right: The agents' network using the regularized joint-rational version of the network. The same activation is used across the layers. The parameters of the rational activation (in orange) function are shared. In both graphs, operations are placed in the grey boxes and parameters in the blue ones, (or orange for the rationals' ones).