JUST STACK IT: DEEPER REINFORCEMENT LEARNING WITH NEAR-ISOMETRIC LAYERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep networks have driven progress in vision and language, yet reinforcement learning (RL) agents still rely on networks a few layers deep, because adding layers degrades their performance. We show that this failure is linked to a loss of conditioning: during training, the Jacobian of a deep network's hidden layers moves away from an isometry, its features lose rank, and the deviations of the layers multiply with depth. Under the non-stationary, bootstrapped objective of RL, this degradation impairs the fitting of later targets. The distance of the Jacobian from an isometry and the rank of the features track performance across architectures and optimizers. We propose a simple residual layer that keeps deep networks close to an isometry. With it, deep networks train stably across value-based, actor-critic and goal-conditioned algorithms and optimizers, and their performance improves or remains stable with depth: they outperform the standard shallow networks and recent deep architectures at matched depth and parameters.

1 INTRODUCTION

Increasing network depth is one of the most reliable ways to improve performance in supervised learning: convolutional networks and Transformers routinely stack dozens of layers (He et al., 2016; Vaswani et al., 2017), and performance improves predictably with model size (Kaplan et al., 2020). Deep reinforcement learning (RL) has largely not followed this trend. Its main algorithms still use networks with a few hidden layers (Schulman et al., 2017; Haarnoja et al., 2018a; Gallici et al., 2025), and adding layers degrades their performance, often to the point of collapse (Figure 1; Bjorck et al., 2021; Ota et al., 2021; Castanyer et al., 2025).

A central difficulty is that the RL objective is non-stationary. Value targets are bootstrapped from the network's own, changing predictions, and the data distribution shifts with the policy (Van Hasselt et al., 2018). Networks trained this way gradually lose the ability to fit new targets, with symptoms such as dormant units, growing weight norms, collapsing representation rank and vanishing gradients (Sokar et al., 2023; Lyle et al., 2025; Kumar et al., 2021; Castanyer et al., 2025). Every added layer is one more transformation whose conditioning can deteriorate as the targets change, so a deep network must remain well conditioned throughout training, not only at initialization.

Recent work shows that larger networks can improve RL agents. SimBa and SimbaV2 combine residual blocks with normalization of inputs, features and weights (Lee et al., 2025a;b), Castanyer et al. (2025) stabilize gradients with multi-skip connections and a Kronecker-factored optimizer, and Wang et al. (2025) train networks with many layers for goal-conditioned RL, but require a self-supervised contrastive objective to do so. Each of these methods combines several changes, such as specialized optimizers, weight projections, distributional objectives or a particular learning algorithm, which makes it difficult to identify what enables depth and whether it transfers to other algorithms. We ask instead which property a network needs so that depth does not harm performance in RL, independently of the algorithm and the optimizer.

We argue that this property is dynamical isometry: networks whose Jacobian singular values stay close to one, so that signals and gradients are neither amplified nor suppressed in any direction, can be trained at large depth (Saxe et al., 2014; Pennington et al., 2017; Schoenholz et al., 2017; Xiao et al., 2018; Martens et al., 2021). We show that deep networks move away from isometry during training and that the deviations of individual layers multiply with depth. RL is sensitive to this degradation throughout training, for two reasons: later targets can have errors along directions that

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

earlier training has suppressed, and each gradient step changes the bootstrapped targets as well as the predictions. Guided by the tangent kernel, we construct a layer whose closeness to an isometry is invariant to the scale of its weights: a fixed orthogonal skip connection plus a residual branch with a single weight matrix, a parameter-free normalization and a shifted ReLU, added with weight $1/K$ in a network of K layers. Building on work on signal propagation, normalization and depth scaling (Zhang et al., 2019; Bachlechner et al., 2021; Zhang & Sennrich, 2019; Dey et al., 2025), we prove that the network's distance from an isometry is bounded by a mean over its layers, not a product.

We evaluate the layer with value-based, actor-critic and goal-conditioned algorithms, with first-order and preconditioned optimizers (RAdam, Muon and PSGD-Kron), and in discrete and continuous control. With PQN and PPO on Atari (Bellemare et al., 2013) and SAC and PPO on MuJoCo Playground (Zakka et al., 2025), performance improves or remains stable with depth, and our networks outperform both the published shallow networks and, at matched depth and parameter count, SimbaV2 and the multi-skip network of Castanyer et al. (2025). The layer also improves TD3 (Fujimoto et al., 2018) on DeepMind Control and PQN on Craftax (Matthews et al., 2024). In goal-conditioned RL, where Wang et al. (2025) found that temporal-difference methods do not benefit from depth, SAC with hindsight relabeling (Andrychowicz et al., 2017) benefits from depth when its critic uses our layer, and outperforms their contrastive agents on most tasks. Our contributions are:

- **A diagnosis of how depth fails in RL.** Through the tangent kernel, we relate the failure to two properties, the isometry of the hidden layers and the rank of the features, which degrade in training, and show that depth multiplies the deviation of each layer.
- **A layer with a depth-independent bound.** A simple residual layer whose deviation from an isometry has a bound that is independent of depth and of the scale of the weights, and depends only on how the weights act on the hidden states.
- **Gains from depth.** Performance improves with depth in most settings and remains stable in the others, across algorithms and optimizers, and exceeds recent RL architectures at matched depth and parameter count.
- **Evidence for the mechanism.** Across all networks we trained, the isometry of the hidden layers and the feature rank track performance, and fixed skip connections of controlled conditioning show that performance depends on the conditioning of the skip path.

2 Background and Related Work

Reinforcement learning and non-stationarity. An agent in a Markov decision process with rewards $r(s, a)$ and discount factor $\gamma \in [0, 1)$ seeks a policy that maximizes the expected discounted return (Sutton & Barto, 2018). Value-based and actor-critic methods learn value functions by *bootstrapping*: the network Q_θ is regressed onto targets computed from its own predictions,

$$\mathcal{L}_t(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_t}\Big[\big(Q_\theta(s,a) - y_t\big)^2\Big], \qquad y_t = r + \gamma \max_{a'} Q_{\bar{\theta}_t}(s', a'), \tag{1}$$

where $\bar{\theta}_t$ is a current or lagged copy of the parameters and $\mathcal{D}_t$ the data collected by recent policies (Mnih et al., 2015). PQN (Gallici et al., 2025) regresses onto λ-returns computed on fresh data from many parallel environments, without a replay buffer or target network. SAC (Haarnoja et al., 2018a) and TD3 (Fujimoto et al., 2018) train critics off-policy on targets from a slowly updated copy, and PPO (Schulman et al., 2017) fits a value function to bootstrapped returns that weight its policy update (Appendix A). In all of them both sides of the regression are non-stationary: the targets y_t change with every update and the inputs $\mathcal{D}_t$ with the policy. Training is a sequence of related regression problems, and a network that has fit the earlier ones must remain able to fit the later ones. Networks often lose this ability, which is known as loss of plasticity (Nikishin et al., 2022; Lyle et al., 2023; Dohare et al., 2024). Proposed methods reset parameters or constrain the weights or their spectra (Kumar et al., 2025; Lewandowski et al., 2025; Lyle et al., 2024), and Rosseau et al. (2026) relate the loss to the conditioning of the tangent kernel.

Signal propagation. For hidden layers $\boldsymbol{h}_k = F_k(\boldsymbol{h}_{k-1}) \in \mathbb{R}^n$, $k = 1, \dots, K$, with layer Jacobians $\boldsymbol{J}_k = \partial \boldsymbol{h}_k / \partial \boldsymbol{h}_{k-1}$, a perturbation of the input propagates forward and the gradient of a loss $\mathcal{L}$ backward through products of these Jacobians:

$$\delta \boldsymbol{h}_K = \boldsymbol{J}_K \boldsymbol{J}_{K-1} \cdots \boldsymbol{J}_1 \, \delta \boldsymbol{h}_0, \qquad \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_k} = \big(\boldsymbol{J}_K \boldsymbol{J}_{K-1} \cdots \boldsymbol{J}_{k+1}\big)^\top \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_K}. \tag{2}$$

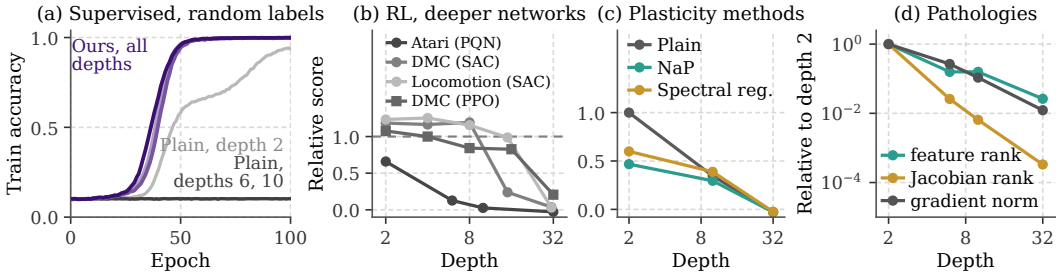

Figure 1: **Deep networks fail in supervised learning and in RL. (a)** CIFAR-10 with fixed random labels: train accuracy of plain networks (grey, darker is deeper) and of ours (purple) at depths 2, 6 and 10. **(b)** Plain networks with PQN on Atari-10, SAC on DeepMind Control and Locomotion, and PPO on DeepMind Control, relative to the baseline (dashed). **(c)** Plain PQN on Qbert and Phoenix, alone and with two interventions against loss of plasticity. **(d)** Plain PQN on Atari-10: feature rank, numerical rank of the Jacobian of the hidden layers, gradient norm. (c), (d) relative to depth 2.

For a plain layer $\boldsymbol{h}_k = \phi(\boldsymbol{W}_k \boldsymbol{h}_{k-1})$, $\boldsymbol{J}_k = \boldsymbol{D}_k \boldsymbol{W}_k$ with $\boldsymbol{D}_k = \mathrm{diag}(\phi'(\boldsymbol{W}_k \boldsymbol{h}_{k-1}))$, so a ReLU zeroes the rows of inactive units. If the singular values of the product are far from one, signals and gradients vanish or explode with depth (Bengio et al., 1994; Glorot & Bengio, 2010). Variance-preserving initializations keep the mean square of the singular values at one (Glorot & Bengio, 2010; He et al., 2015), but with Gaussian weights their spread still grows with depth. A network is *dynamically isometric* when all singular values are close to one (Saxe et al., 2014; Pennington et al., 2017). Orthogonal weights achieve this at initialization for activations that are close to linear, whereas ReLU discards the directions of inactive units and saturating activations are isometric only in their linear regime (Pennington et al., 2018). With suitable initialization and activations, networks with thousands of layers become trainable (Xiao et al., 2018; Martens et al., 2021). Residual networks approach isometry by down-weighting their branches with depth (Zhang et al., 2019; De & Smith, 2020; Hayou et al., 2021): a factor $1/\sqrt{K}$ keeps the output of a random network bounded and lets hyperparameters transfer across depths (Yang et al., 2024; Bordelon et al., 2024), and CompleteP (Dey et al., 2025) shows that $1/K$ is the only scaling under which every layer keeps learning features as depth grows. These results concern networks at or near initialization.

Scaling networks in RL. Spectral normalization bounds the largest singular value of every weight matrix, leaving the smallest unconstrained, and stabilizes deep critics (Bjorck et al., 2021). SimBa (Lee et al., 2025a) scales actor-critic networks with residual blocks that keep a linear path from input to output, pre-block layer normalization and running normalization of the inputs, and attributes its gains to a bias toward simple functions. SimbaV2 (Lee et al., 2025b) additionally constrains weights and features to the unit hypersphere, which keeps the effective learning rate constant, and uses a distributional critic and reward scaling. The multi-skip network of Castanyer et al. (2025) adds the encoder output to the input of every block and is trained with PSGD-Kron (Li, 2018). BRO (Nauman et al., 2024) and BBF (Schwarzer et al., 2023) scale mainly the width, with resets and regularization, and SimBa outperforms BRO (Lee et al., 2025a), while the gains of mixtures of experts (Obando-Ceron et al., 2024) were traced largely to the tokenization of the encoder output (Sokar et al., 2025). We therefore compare with SimBa, SimbaV2, the multi-skip network and the contrastive agents of Wang et al. (2025). Appendix B discusses further related work.

3 How Depth Fails in Reinforcement Learning

3.1 Deep networks lose their conditioning in training

We first train *plain* networks, without skip connections, on a supervised task, following Castanyer et al. (2025): a convolutional encoder followed by fully connected layers, on CIFAR-10 with fixed random labels (Figure 1a). At depth 2 the network fits the labels slowly, whereas at depths 6 and 10 the features collapse to rank one within six epochs, and by the end of training almost all singular values of the Jacobian of the hidden layers are below 10^{-3}. We call these directions *lost*, since signals and gradients along them are suppressed by three orders of magnitude or more. Our network (Section 4) fits the labels at every depth. With changing labels, the failure is more severe: with labels

reshuffled every 20 epochs, the deepest plain network never recovers from the first reshuffle, and a fully connected plain network of depth 32 fits six random labellings in a row and then stays at chance, although trained for as long on the true labels it reaches a train accuracy of 0.83 (Appendix K).

In RL, both the targets and the inputs of the regression are non-stationary. We call the published network of an agent the *baseline* and the same network with more hidden layers the *plain* network. The plain networks of PQN and SAC lose performance with depth on every benchmark we tried (Figure 1b). Two interventions designed against loss of plasticity do not prevent this. On Qbert and Phoenix, normalize-and-project (Lyle et al., 2024) and spectral regularization (Lewandowski et al., 2025), at learning rates from 0.1 to 3 times the published one, lower the score at depth 2 and leave the network collapsed at depth 32 at every learning rate (Figure 1c; Appendix J). Spectral regularization holds the largest singular value of every weight matrix between 1.0 and 1.2, and the network still collapses: bounding the largest singular value of each weight matrix does not keep the smallest singular values of the product of the layer Jacobians away from zero.

The deepened networks show pathologies reported separately in the literature (Figure 1d). On Atari, the rank of the features that reach the output layer falls from 38 at depth 2 to 1 at depth 32, where all inputs map to the same features. The Jacobian of the hidden layers keeps 30% of its directions at depth 2 and none at depth 32, and the gradient that reaches each layer shrinks 60-fold, as Castanyer et al. (2025) observed. The fraction of dormant units (Sokar et al., 2023) rises from 0.52 to 0.82 up to depth 10 but falls to 0.27 at depth 32, where the units are active but constant across inputs, so counting inactive units does not detect collapsed features. Since these quantities change together within the plain networks, this experiment cannot tell which of them matters. Section 3.2 relates them to two properties, and Section 5.2 compares them across architectures and optimizers.

3.2 A TANGENT KERNEL VIEW OF DEPTH

Let $X = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_m)$ be a batch of inputs and $f_\theta(X) \in \mathbb{R}^m$ the predictions on them, with parameters $\theta \in \mathbb{R}^p$. To first order, a gradient step with learning rate η on a loss $\mathcal{L}_t$ changes them by

$$\Delta f_\theta(X) = -\eta\, \boldsymbol{K}\, \nabla_f \mathcal{L}_t, \qquad \boldsymbol{K} = \boldsymbol{J}_\theta \boldsymbol{J}_\theta^\top \in \mathbb{R}^{m\times m}, \qquad \boldsymbol{J}_\theta = \frac{\partial f_\theta(X)}{\partial \theta} \in \mathbb{R}^{m\times p}, \tag{3}$$

where $\boldsymbol{K}$ is the empirical neural tangent kernel (Jacot et al., 2018). The error $\nabla_f \mathcal{L}_t$, for Equation 1 the temporal-difference error, is determined by the targets, and the kernel determines how the network responds to it. The component of the error along an eigenvector of $\boldsymbol{K}$ decays at a rate proportional to its eigenvalue, so errors along directions with small eigenvalues are fitted slowly. Under a fixed objective, training aligns the leading eigenvectors of the kernel with the target (Baratin et al., 2021; Fort et al., 2020; Atanasov et al., 2022). Under a non-stationary objective, later targets can have errors along directions with small eigenvalues, and the targets change before these errors are reduced. Bootstrapping adds a second dependence on the kernel. The targets are computed from the network's own predictions, so a gradient step changes the targets as well as the predictions. For targets computed with the current parameters, as in PQN, the temporal-difference error changes by $-\eta\,(\boldsymbol{K} - \gamma \boldsymbol{K}')\nabla_f \mathcal{L}_t$, where $\boldsymbol{K}'$ is the kernel between the next and the current inputs (Achiam et al., 2019; Yue et al., 2023). Its entry $\nabla_\theta f_\theta(\boldsymbol{x}'_i)^\top \nabla_\theta f_\theta(\boldsymbol{x}_j)$ measures how much an update on $\boldsymbol{x}_j$ changes the prediction at the next input $\boldsymbol{x}'_i$, and thereby the target of $\boldsymbol{x}_i$. With a target network, the targets change with a delay. Temporal-difference learning can make the features of an input and of its successor similar (Kumar et al., 2022). The gradients at the two inputs are then nearly equal, so that $\boldsymbol{K}' \approx \boldsymbol{K}$ and the error decreases at $1-\gamma$ times the rate of a fixed target.

The kernel decomposes over layers. With pre-activations $\boldsymbol{z}_k = \boldsymbol{W}_k \boldsymbol{h}_{k-1}$ and a scalar output, the gradient with respect to $\boldsymbol{W}_k$ is the outer product of the backward signal $\boldsymbol{b}_k = \partial f_\theta / \partial \boldsymbol{z}_k$ and the activation $\boldsymbol{h}_{k-1}$, so that

$$\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_k \underbrace{\big\langle \boldsymbol{h}_{k-1}(\boldsymbol{x}_i),\, \boldsymbol{h}_{k-1}(\boldsymbol{x}_j) \big\rangle}_{\text{features entering layer } k} \underbrace{\big\langle \boldsymbol{b}_k(\boldsymbol{x}_i),\, \boldsymbol{b}_k(\boldsymbol{x}_j) \big\rangle}_{\text{errors reaching layer } k}. \tag{4}$$

Each layer contributes the similarity of the features that enter it times the similarity of the errors that reach it from the output. By Equation 2, both are determined by products of layer Jacobians: differences between inputs reach layer k through $\boldsymbol{J}_{k-1} \cdots \boldsymbol{J}_1$, and errors through $(\boldsymbol{J}_K \cdots \boldsymbol{J}_{k+1})^\top$. Small singular values make distinct inputs indistinguishable and errors vanish, large ones separate

nearby inputs and make errors grow, and a spread of singular values amplifies some directions and suppresses others (Bengio et al., 1994; Schoenholz et al., 2017; Pennington et al., 2017). All three are excluded when every layer is close to an isometry. The feature term must in addition keep its rank, since features that collapse onto a few directions leave the kernel with few nonzero eigenvalues (Kumar et al., 2021; Lyle et al., 2022). The kernel therefore suggests two requirements for a deep network: the closeness of the Jacobian of its hidden layers to an isometry, and the rank of its features.

Each of the pathologies of Section 3.1 affects part of these two quantities. Dormant units set rows of $\boldsymbol{D}_k$ to zero, feature-rank collapse degrades the feature term, and growing or ill-conditioned weights distort the spectrum of the Jacobian or, under normalization, reduce the effective learning rate (Lyle et al., 2024). Lyle et al. (2023) find that none of them explains loss of plasticity on its own. The gradient norm measures neither quantity directly: it scales with the size of the remaining error, and it depends on the backward Jacobian only through $\|\boldsymbol{J}\|_F^2 = \sum_i \sigma_i^2$, not through the spread of the singular values. It is small both when a network has collapsed and when it has fitted its targets.

Depth makes the requirement stricter. If the singular values of each layer Jacobian lie within $1 \pm \delta$, those of the product over K layers lie within $(1 \pm \delta)^K$. When the deviations of the layers are independent, as for random weights, they partially cancel, and $\delta \propto 1/\sqrt{K}$ keeps the product bounded, which is the regime of initialization schemes and residual scaling rules (Xiao et al., 2018; Hayou et al., 2021; Yang et al., 2024). Training can correlate the layers. When their deviations align, the product reaches this bound, and with $\delta = c/\sqrt{K}$ it grows as $(1 + c/\sqrt{K})^K \approx e^{c\sqrt{K}}$, so that only $\delta \propto 1/K$ keeps the product bounded for arbitrary weights. In our experiments at depth 32, a $1/\sqrt{K}$ multiplier scores 0.83 on Atari, against 0.98 for $1/K$ (Appendix J). Under a non-stationary objective, the condition must hold for the trained weights throughout training, which neither isometry at initialization nor constraints on each weight matrix (Figure 1c) provide.

4 A Near-Isometric Layer

Because bootstrapped targets change throughout training, the condition of Section 3.2 must hold for the trained weights. We therefore seek a layer whose closeness to an isometry depends neither on the scale of its weights nor on depth. A residual layer $\boldsymbol{h}_k = \boldsymbol{S}_k \boldsymbol{h}_{k-1} + \varepsilon\, g_k(\boldsymbol{h}_{k-1})$ with a *fixed* orthogonal matrix $\boldsymbol{S}_k$, in the simplest case the identity, contains a path that is an exact isometry and that no update can change. The sum of two isometries is in general far from an isometry (Appendix C), but the sum of an isometry and a small perturbation remains close to an isometry: by Weyl's inequality, the singular values of $\boldsymbol{S}_k + \varepsilon \boldsymbol{G}_k$, with $\boldsymbol{G}_k$ the Jacobian of the branch, lie within $\varepsilon \|\boldsymbol{G}_k\|_2$ of one. Over K layers these deviations accumulate, and by Section 3.2 only $\varepsilon \propto 1/K$ keeps the product bounded for arbitrary weights. Our layer is

$$\boldsymbol{h}_k = \boldsymbol{S}_k \boldsymbol{h}_{k-1} + \frac{1}{K}\, \phi_{\boldsymbol{b}_k}\big(\nu(\boldsymbol{W}_k \boldsymbol{h}_{k-1})\big), \qquad \nu(\boldsymbol{z}) = \frac{\boldsymbol{z}}{\mathrm{rms}(\boldsymbol{z})}, \qquad \phi_{\boldsymbol{b}}(\boldsymbol{u}) = \max(\boldsymbol{u}, \boldsymbol{b}), \tag{5}$$

where $\mathrm{rms}(\boldsymbol{z}) = \|\boldsymbol{z}\|_2/\sqrt{n}$, the maximum is taken elementwise, and $\boldsymbol{S}_k = \boldsymbol{I}$ unless stated otherwise. Each layer has a single weight matrix $\boldsymbol{W}_k \in \mathbb{R}^{n \times n}$, initialized orthogonally, so every weight matrix of the hidden layers lies inside a residual branch. The layers are stacked between a linear input layer, followed by ν, and a linear output layer, and the *depth* is the number of layers K.

The multiplier $1/K$ is the one that CompleteP (Dey et al., 2025) derives for Transformers from a different requirement, that every layer keeps learning features as depth grows. The multiplier acts on a branch of fixed scale next to a fixed isometric skip connection, as the bound below requires, and it is applied at every step rather than only at initialization: initializing the branch weights at scale $1/K$ would not suffice, because the weights grow during training, whereas the multiplier stays fixed. As in CompleteP, we use one learning rate at all depths (Section 5.1).

The multiplier bounds the branch only if the branch itself has a fixed scale, which is the role of the normalization ν: its output has unit RMS for any weights, and its Jacobian, $(\boldsymbol{I} - \boldsymbol{u}\boldsymbol{u}^\top/n)/\mathrm{rms}(\boldsymbol{z})$ with $\boldsymbol{u} = \nu(\boldsymbol{z})$, is a scaled isometry on all but one direction. The branch is therefore invariant to the scale of $\boldsymbol{W}_k$, so the growth of weight norms and target magnitudes during RL training (Lyle et al., 2025) cannot enlarge it, and the scale of the value function is set by the linear output layer. LayerNorm performs on par (Appendix J); we use the simplest form, the one Proposition 1 uses. The shifted ReLU of Qi et al. (2020) moves ReLU towards an isometry: its Jacobian is the identity

on every unit above the threshold. With normalized pre-activations of unit scale, the thresholds $\boldsymbol{b}_k$ have a fixed interpretation: -1 passes about five inputs in six unchanged where ReLU passes half. Each unit learns its threshold, and thus the fraction of inputs it passes unchanged.

Proposition 1. *Let $T = F_K \circ \cdots \circ F_1$ be the composition of K layers (Equation 5) with arbitrary weights, let $c_k = \|\boldsymbol{W}_k\|_2 / \mathrm{rms}(\boldsymbol{W}_k \boldsymbol{h}_{k-1})$ and $\bar{c} = \frac{1}{K}\sum_k c_k$, and assume $c_k \le K/2$. Wherever T is differentiable, every singular value σ of its Jacobian satisfies*

$$\exp\Big(-\bar{c}\,\big(1 + \max_k c_k/K\big)\Big) \;\le\; \sigma \;\le\; \exp\big(\bar{c}\big). \tag{6}$$

Every c_k is invariant to the scale of $\boldsymbol{W}_k$, and $c_k = 1/\mathrm{rms}(\boldsymbol{h}_{k-1})$ if $\boldsymbol{W}_k$ is orthogonal. If moreover $\boldsymbol{b}_k \le 0$ for all k, then $\mathrm{rms}(\boldsymbol{h}_K) \le \mathrm{rms}(\boldsymbol{h}_0) + 1$.

The proof is in Appendix C. The constant c_k is the ratio between the largest gain of $\boldsymbol{W}_k$, its spectral norm, and its gain on the actual hidden state. It is close to one for an orthogonal $\boldsymbol{W}_k$ and a normalized input, and it is large only when the hidden state has little component along the directions in which $\boldsymbol{W}_k$ has its largest singular values. The bound does not depend on depth, and it enters through the mean $\bar{c}$ over layers, whereas the condition numbers of the layers of a plain network multiply. It is a conditional bound: it implies closeness to an isometry only while the constants c_k remain small, which the architecture does not enforce and which we therefore measure. On Atari, the constants remain small during training: at depth 32, $\max_k c_k/K$ ends at 0.10 or less on average under every optimizer and at 0.23 at most in any run, so the singular values of every layer stay close to one, and the measured distance from an isometry is at most a sixth of what the bound allows (Appendix I).

If $c_k \le K/2$ holds on a convex set that contains the hidden states, the same factors bound the ratio of the distances between the representations of two inputs after and before the hidden layers. The hidden layers are then bi-Lipschitz: they cannot map distinct inputs onto each other, and because they preserve all pairwise distances up to these factors, they also preserve the total variance of the features, so the features cannot collapse as in the plain networks of Section 3.1 (Appendix C.3). An identity path alone does not give this, since an unscaled branch can cancel or dominate it. The multi-skip network of Castanyer et al. (2025) keeps an identity path through every block, but its branches are not scaled with depth and it adds the encoder output back at every block, so that the hidden state and its Jacobian grow with depth. Projecting the representation onto the unit sphere after every block, as in SimbaV2, removes its norm. Bounding the distortion requires both a branch of fixed scale and the weight $1/K$. The rank of the features, which the bound does not control directly, also stays high in our networks at every depth and along the hidden layers (Section 5.2 and Appendix I).

Architecture and optimizer. With a preconditioner $\boldsymbol{P}$, the update of Equation 3 becomes $\Delta f_\theta(X) = -\eta\, \boldsymbol{J}_\theta \boldsymbol{P} \boldsymbol{J}_\theta^\top \nabla_f \mathcal{L}_t$: the architecture determines $\boldsymbol{J}_\theta$ and the optimizer $\boldsymbol{P}$. For the squared error of Equation 1, the Gauss-Newton approximation of the Hessian that second-order methods use, $\boldsymbol{J}_\theta^\top \boldsymbol{J}_\theta$, has the same nonzero eigenvalues as the kernel of Section 3.2 (Appendix D.1). We use three optimizers that approximate it differently (Appendix D). RAdam (Liu et al., 2020), the default of PQN, rescales each parameter by a diagonal estimate. PSGD-Kron (Li, 2018), Kron for short, which Castanyer et al. (2025) pair with the multi-skip network, whitens the activations and the backward signals of each layer with two Kronecker factors, as K-FAC and Shampoo do (Martens & Grosse, 2015; Gupta et al., 2018). Muon (Jordan et al., 2024) replaces the update of each weight matrix by its orthogonal polar factor, whose singular values all equal one. In language and vision models it yields flatter weight spectra and features of higher rank than Adam (Liu et al., 2025; Ruan et al., 2026), and to our knowledge it has not been used for deep RL agents trained from scratch. A preconditioner can rescale the directions that the Jacobians preserve, but it cannot restore a lost direction, since the gradient has no component along it. We therefore expect a near-isometric network to train at every depth under any optimizer, a network whose Jacobians lose directions to fail under every optimizer, and one whose Jacobians keep their directions but distort their spectrum to be helped by a whitening preconditioner and still degrade with depth.

5 Experiments

5.1 Performance improves with depth or remains stable

We use PQN on Atari-10 (Aitchison et al., 2023), SAC on DeepMind Control (Tassa et al., 2018) and on the Locomotion tasks of MuJoCo Playground (Zakka et al., 2025), PPO on DeepMind Con-

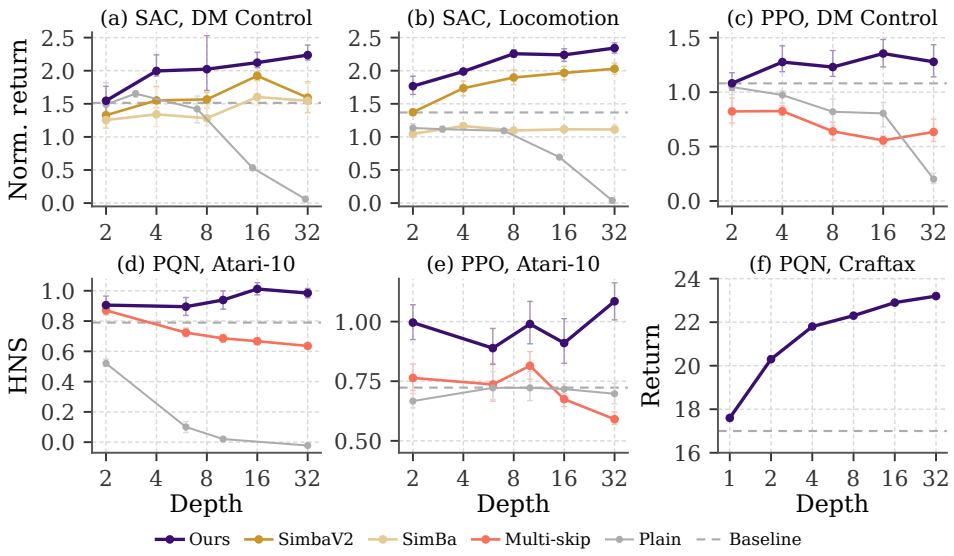

Figure 2: **Performance against depth.** IQM with 95% confidence intervals. Dashed: the baseline under its published optimizer, in (a) and (b) with the training improvements of SimbaV2. Grey: the plain network with the published settings. **(a)–(c)** Returns relative to the published agent. **(d), (e)** Human-normalized scores (HNS); ours under Muon (d) and Kron (e), multi-skip under Kron (other optimizers: Figure 4). **(f)** Return on Craftax-Symbolic, mean of three runs.

trol and Atari-10, TD3 on DeepMind Control, PQN on Craftax-Symbolic, and SAC with hindsight relabeling on JaxGCRL (Bortkiewicz et al., 2025) (Appendix E). Depth is the number of weight matrices between the input and the output layer. A block of SimbaV2, SimBa or the multi-skip network contains two of them, and all compared networks have the same number of parameters at every depth (Appendix F). As in Section 3.1, the *baseline* is the published shallow network and the *plain* network its deepened version. Our layers form the hidden layers of the networks trained by bootstrapping: the Q-network of PQN, the critics of SAC and TD3, and the value network of PPO (on Atari, a value head on the shared encoder). The policy networks keep their published architecture, since deepening them did not improve the score (Appendix F). Prior networks use their published settings, and we sweep the learning rate where none is published. Our network uses one setting per benchmark, chosen at depth 32 and used at every depth, since tuning at shallower depths barely changes the scores; for PQN under RAdam and goal-conditioned SAC the published settings carried over unchanged (Appendix G). On DeepMind Control we report Adam, the optimizer of the baselines and SimbaV2; Kron scores similarly at a higher cost. Scores are the interquartile mean (IQM) over pooled runs of three seeds with stratified bootstrap 95% confidence intervals (Agarwal et al., 2021): human-normalized scores on Atari and normalized returns on Playground.

Figure 2 shows how performance changes with depth for the best configuration of each method. At depth 32, our network scores highest on every benchmark, with confidence intervals disjoint from those of the baseline and of the prior deep network on Atari with PQN and PPO, with SAC on DeepMind Control and Locomotion, and with TD3 (Table 6 in Appendix H). Appendix H also gives every optimizer (Figures 4 and 6), TD3, Craftax and the T1 humanoid (Figure 5), and per-task results. With SAC, our network improves with depth and scores above SimbaV2 and SimBa at every depth, with non-overlapping intervals on Locomotion. SimbaV2, evaluated up to depth 16 by Lee et al. (2025b), declines at depth 32 on DeepMind Control (1.59 against 1.92). A wider SimbaV2 network with the same 8.4M parameters scores 1.93 on DeepMind Control, against 2.24 for ours. At the same parameter count, shallower and wider networks of our layers score lower than ours at depth 32, slightly at depth 8 and clearly at depth 2 (Appendix H): with our layer, depth is at least as efficient a way to add capacity as width. With PPO, a value network of our layers improves on the baseline at every depth from 4 on, whereas the multi-skip network stays below it at every depth.

On Atari with PQN (Figure 2d; every optimizer in Figure 4), the results follow the predictions of Section 4. The plain network, whose Jacobians lose directions, degrades with depth under every optimizer, to about zero at depth 32. The multi-skip network, whose Jacobians keep their directions

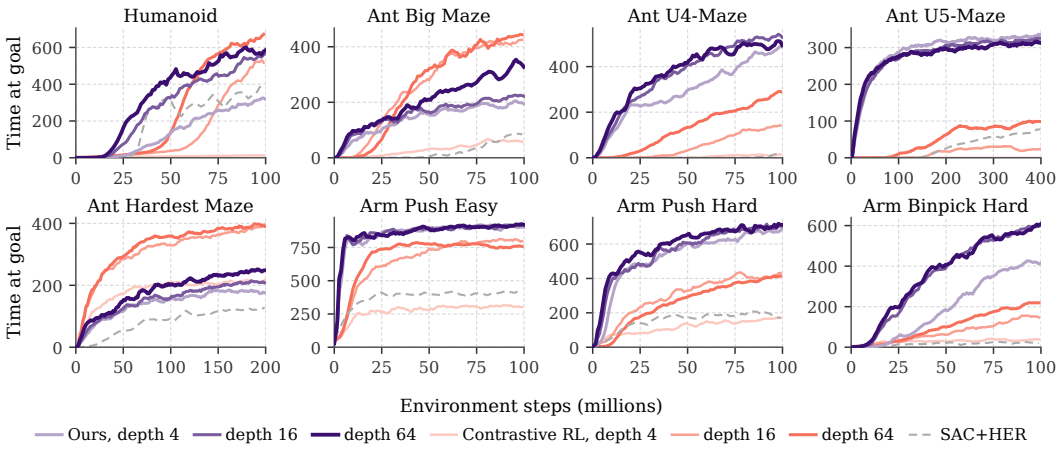

Figure 3: **Goal-conditioned RL** on the eight JaxGCRL tasks of Wang et al. (2025): time at goal (out of 1,000) against environment steps. Ours: SAC with hindsight relabeling and our critic, mean over three seeds; contrastive RL and SAC+HER: their published results.

but distort their spectrum, scores higher under Kron than under RAdam but degrades with depth under both, and stays below the baseline under the same optimizer at every depth. Castanyer et al. (2025) report gains relative to a plain network of the same depth. Their shallowest network already has three more hidden layers than the published one, and in our runs a plain network at depth 2 scores well below the baseline (0.52 against 0.79 under RAdam). Their largest network has depth 10. Our network is stable or improves with depth under every optimizer. Behind the deeper Impala encoder, our network at depth 16 also scores above the published and multi-skip networks under all three optimizers (Appendix H). With PPO on Atari (Figure 2e), our value network scores above the baseline at every depth under every optimizer, while the multi-skip network declines past depth 10.

In goal-conditioned RL, Wang et al. (2025) found that SAC with hindsight relabeling does not benefit from depth and scaled a self-supervised contrastive agent to a thousand layers instead. With our layers in its critic, and with their data and learning rate, SAC with hindsight relabeling improves with depth on four of the eight tasks and changes by less than 10% on the others, and at its best depth it scores above their SAC agent on every task. At the best depth of each method, it scores higher than their contrastive agent on five of the eight tasks, within 10% on one and lower on two (Figure 3). Our network was not tuned for these tasks, and tuning may improve it further. On Craftax-Symbolic with PQN, the return of our network rises with depth, from 17.6 at depth 1 to 23.2 at depth 32 under the same settings, against 17.0 for the baseline, and at depth 32 it collects iron, enters the dungeon and defeats orc mages more often than the baseline (Figure 2f). With TD3, our network scores above the baseline at every depth and doubles it at depth 32 (2.05 against 1.00; Figure 5).

5.2 Jacobian conditioning tracks performance

At the end of training, we measure the Jacobian of the hidden layers of the networks of Section 5.1 (three seeds; Appendix I). Its distance from an isometry is $d = \left(\frac{1}{n}\sum_i (\log \sigma_i)^2\right)^{1/2}$, which is zero for an isometry and combines the overall gain with the spread of the singular values σ_i. Singular values below 10^{-3} count as lost and are set to 10^{-3}. We also report the share of singular values within a factor of two of one and the feature rank (Kumar et al., 2021).

In our network, no singular value of this Jacobian is lost at any depth or on any benchmark, and its distance from an isometry is 0.22 to 0.42 on Atari and 0.5 to 1.9 with SAC and PPO, whereas the baselines and the plain networks lose 70% to all of their singular values (Figure 12 in Appendix I). The multi-skip network keeps its directions on Atari but amplifies every signal by about $e^{2.5}$, and with PPO it loses 95% of them already at depth 2. SimbaV2 is as close to an isometry as our network with SAC, but its features have lower rank (88 to 95 against 108 to 114): at equal parameter count its residual path has half our width, a likely reason it scores lower.

Across all networks we trained, the Jacobian measures and the effective rank of the tangent kernel in Table 1 track the score on every benchmark, and the feature rank does so on Atari and with SAC.

Table 1: **Which quantities track performance.** Spearman correlation between each quantity at the end of training and the final score across all runs of all networks and optimizers (1,565 runs), within each game or task, averaged. Jacobian and tangent kernel of the hidden layers; share: share of singular values within a factor of two of one. Bold: 95% bootstrap interval excludes zero.

	Jacobian		Tangent kernel		Features, weights, loss and gradient				
	Share	Distance from isometry	Effective rank	Trace	Feature rank	Dormant units	Weight norm	Largest Hessian eigenvalue	Gradient norm
Atari, PQN	$\mathbf{+0.61}$	$\mathbf{-0.53}$	$\mathbf{+0.32}$	-0.01	$\mathbf{+0.53}$	$\mathbf{-0.14}$	$\mathbf{+0.30}$	$\mathbf{+0.14}$	$+0.05$
DMC, PPO	$\mathbf{+0.32}$	$\mathbf{-0.34}$	$\mathbf{+0.39}$	-0.08	$+0.18$	$\mathbf{-0.25}$	$\mathbf{+0.25}$	–	$+0.01$
DMC, SAC	$\mathbf{+0.27}$	$\mathbf{-0.26}$	$\mathbf{+0.26}$	-0.03	$\mathbf{+0.42}$	$\mathbf{-0.37}$	$\mathbf{+0.24}$	–	-0.07

The relation does not depend on the choice of Jacobian measure: the share of singular values within a factor of two of one gives similar correlations for any factor between 1.5 and 4 (Appendix I). Castanyer et al. (2025) attribute the failure of deep networks to vanishing gradients and design their network to keep the gradient norm from decaying. The gradient norm does fall as a network collapses (Figure 1d), but it measures neither the isometry of the hidden layers nor the rank of the features (Section 3.2), and its magnitude depends on the architecture and the optimizer: under Kron the multi-skip network has a four times smaller gradient than under Muon and a higher score. Across all networks it does not track the score, and neither does the trace of the tangent kernel. The fraction of dormant units (Sokar et al., 2023) has the expected sign but is weak on Atari, the weight norm correlates positively, opposite to its role as a symptom of lost plasticity, and the largest Hessian eigenvalue correlates weakly positively. The simplicity score at initialization of Lee et al. (2025a) is the same for six variants of our layer whose scores range from 0.48 to 0.98 (Figure 14, Appendix I).

To test this more directly, we vary only the conditioning of the skip path: at depth 32 on Atari, the identity skip of every layer is replaced by a fixed matrix that is never trained, all else unchanged. A random orthogonal matrix, which is an exact isometry, performs like the identity, whereas every non-isometric fixed matrix lowers the score, by 7% to 27% under Muon and 14% to 17% under RAdam (Appendix J). Since the fixed matrix is never trained, this is not an effect of capacity. Under Kron, which whitens the signals these matrices distort, they lower the score much less, but Kron cannot restore lost directions: without the skip, the network fails under every optimizer.

Ablations. Each of the four components of the layer contributes at depth 32 under Muon (Appendix J): without the skip connection the score falls from 0.98 to 0.26, without the $1/K$ multiplier to 0.48 (0.83 with $1/\sqrt{K}$), without the normalization to 0.82, and with ReLU and GELU in place of the shifted ReLU to 0.94 and 0.93.

6 Discussion

Our results indicate that deep networks fail in RL through a loss of conditioning during training: the Jacobian of the hidden layers moves away from an isometry, the features lose rank, and the deviations of the layers multiply with depth. Since the bootstrapped targets of RL keep changing, a network that has lost this conditioning can fit the later targets only poorly. The distance of the Jacobian from an isometry and the rank of the features track performance across networks, optimizers and algorithms. A preconditioner such as Kron can compensate for a distorted spectrum but not for lost directions, so the conditioning has to be maintained by the architecture. We introduced a simple layer whose components follow from the analysis of Section 3.2; it keeps the Jacobian of the hidden layers close to an isometry at every depth and, in our experiments, keeps the rank of the features high. With it, performance improves or remains stable with depth for value-based and actor-critic agents on Atari, DeepMind Control, Locomotion and Craftax, under first-order and preconditioned optimizers, and our deep networks outperform the published shallow networks and recent deep architectures. In goal-conditioned RL, temporal-difference learning then benefits from depth, where it had been found not to (Wang et al., 2025).

Limitations. Deeper networks of our layers can improve performance, but their computational cost also rises with depth (Appendix H.6). Closeness to an isometry could also be imposed on convolutional encoders, recurrent and attention layers, which we leave for future work.

AI USE STATEMENT

We used large language models to write and debug parts of the experiment code, to keep and search the scientific log of the project, and to review and edit the text of this paper. All code was checked by the authors, and all experiments, results and claims were verified by the authors, who take full responsibility for the content of this paper.

REFERENCES

Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C Machado. Loss of plasticity in continual deep reinforcement learning. In *Conference on lifelong learning agents*, pp. 620–636. PMLR, 2023.

Joshua Achiam, Ethan Knight, and Pieter Abbeel. Towards characterizing divergence in deep Q-learning. *arXiv preprint arXiv:1903.08894*, 2019.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Matthew Aitchison, Penny Sweetser, and Marcus Hutter. Atari-5: Distilling the arcade learning environment down to five games. In *International Conference on Machine Learning*, pp. 421–438. PMLR, 2023.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, 2017.

Cem Anil, James Lucas, and Roger Grosse. Sorting out Lipschitz function approximation. In *International conference on machine learning*, pp. 291–301. PMLR, 2019.

Jordan Ash and Ryan P Adams. On warm-starting neural network training. *Advances in neural information processing systems*, 33:3884–3894, 2020.

Alexander Atanasov, Blake Bordelon, and Cengiz Pehlevan. Neural networks as kernel learners: The silent alignment effect. In *International Conference on Learning Representations*, 2022.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W. Cottrell, and Julian McAuley. ReZero is all you need: Fast convergence at large depth. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 1352–1361, 2021.

Aristide Baratin, Thomas George, César Laurent, R. Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. Implicit regularization via neural feature alignment. In *Proceedings of the Twenty-Fourth International Conference on Artificial Intelligence and Statistics*, 2021.

Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, 2019.

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

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

Tudor Berariu, Wojciech Czarnecki, Soham De, Jorg Bornschein, Samuel Smith, Razvan Pascanu, and Claudia Clopath. A study on the plasticity of neural networks. *arXiv preprint arXiv:2106.00042*, 2021.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. *arXiv preprint arXiv:2409.20325*, 2024.

Nils Bjorck, Carla P Gomes, and Kilian Q Weinberger. Towards deeper deep reinforcement learning with spectral normalization. *Advances in neural information processing systems*, 34:8242–8255, 2021.

Blake Bordelon, Lorenzo Noci, Mufan Bill Li, Boris Hanin, and Cengiz Pehlevan. Depthwise hyperparameter transfer in residual networks: Dynamics and scaling limit. In *International Conference on Learning Representations*, 2024.

Michał Bortkiewicz, Władysław Pałucki, Vivek Myers, Tadeusz Dziarmaga, Tomasz Arczewski, Łukasz Kuciński, and Benjamin Eysenbach. Accelerating goal-conditioned reinforcement learning algorithms and research. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2408.11052.

Roger Creus Castanyer, Johan Obando-Ceron, Lu Li, Pierre-Luc Bacon, Glen Berseth, Aaron Courville, and Pablo Samuel Castro. Stable gradients for stable learning at scale in deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2506.15544.

Tong Che. Second-order Muon done right: A principled marriage of spectral geometry and curvature. *arXiv preprint arXiv:2608.09763*, 2026.

Wesley Chung, Lynn Cherif, David Meger, and Doina Precup. Parseval regularization for continual reinforcement learning. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2412.07224.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, pp. 854–863. PMLR, 2017.

Soham De and Samuel L. Smith. Batch normalization biases residual blocks towards the identity function in deep networks. In *Advances in Neural Information Processing Systems*, 2020.

Nolan Dey, Bin Claire Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: CompleteP enables compute-efficient deep transformers. In *Advances in Neural Information Processing Systems*, 2025.

Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, 2024.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M. Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. In *Advances in Neural Information Processing Systems*, 2020.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7IzeL0kflu.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Soufiane Hayou, Eugenio Clerico, Bobby He, George Deligiannidis, Arnaud Doucet, and Judith Rousseau. Stable ResNet. In *Proceedings of the Twenty-Fourth International Conference on Artificial Intelligence and Statistics*, pp. 1324–1332, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G. M. Araújo. CleanRL: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

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

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/. Blog post.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

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

Aviral Kumar, Rishabh Agarwal, Tengyu Ma, Aaron Courville, George Tucker, and Sergey Levine. DR3: Value-based deep reinforcement learning requires explicit regularization. In *International Conference on Learning Representations*, 2022.

Saurabh Kumar, Henrik Marklund, and Benjamin Van Roy. Maintaining plasticity in continual learning via regenerative regularization. In *Proceedings of the 3rd Conference on Lifelong Learning Agents*, volume 274 of *Proceedings of Machine Learning Research*, pp. 410–430, 2025.

Hojoon Lee, Dongyoon Hwang, Donghu Kim, Hyunseung Kim, Jun Jet Tai, Kaushik Subramanian, Peter R. Wurman, Jaegul Choo, Peter Stone, and Takuma Seno. Simba: Simplicity bias for scaling up parameters in deep reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=jXLiDKsuDo.

Hojoon Lee, Youngdo Lee, Takuma Seno, Donghu Kim, Peter Stone, and Jaegul Choo. Hyperspherical normalization for scalable deep reinforcement learning. In *International Conference on Machine Learning*, 2025b. URL https://arxiv.org/abs/2502.15280.

Alex Lewandowski, Michał Bortkiewicz, Saurabh Kumar, András György, Dale Schuurmans, Mateusz Ostaszewski, and Marlos C Machado. Learning continually by spectral regularization. In *International Conference on Learning Representations*, 2025.

Xi-Lin Li. Preconditioned stochastic gradient descent. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5):1454–1466, 2018.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. In *International Conference on Learning Representations*, 2022.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning*, pp. 23190–23211. PMLR, 2023.

Clare Lyle, Zeyu Zheng, Khimya Khetarpal, James Martens, Hado P van Hasselt, Razvan Pascanu, and Will Dabney. Normalization and effective learning rates in reinforcement learning. *Advances in Neural Information Processing Systems*, 37:106440–106473, 2024.

Clare Lyle, Zeyu Zheng, Khimya Khetarpal, Hado van Hasselt, Razvan Pascanu, James Martens, and Will Dabney. Disentangling the causes of plasticity loss in neural networks. In *Proceedings of the 3rd Conference on Lifelong Learning Agents*, volume 274 of *Proceedings of Machine Learning Research*, pp. 750–783, 2025.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International Conference on Machine Learning*, 2015.

James Martens, Andy Ballard, Guillaume Desjardins, Grzegorz Swirszcz, Valentin Dalibard, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Rapid training of deep neural networks without skip connections or normalization layers using deep kernel shaping. *arXiv preprint arXiv:2110.01765*, 2021.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *International Conference on Machine Learning*, 2024.

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

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample-efficient continuous control. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/pdf/2405.16158.

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

Johan Samir Obando-Ceron, Ghada Sokar, Timon Willi, Clare Lyle, Jesse Farebrother, Jakob Nicolaus Foerster, Gintare Karolina Dziugaite, Doina Precup, and Pablo Samuel Castro. Mixtures of experts unlock parameter scaling for deep RL. In *International Conference on Machine Learning*, pp. 38520–38540. PMLR, 2024.

Kei Ota, Devesh K Jha, and Asako Kanezaki. Training larger networks for deep reinforcement learning. *arXiv preprint arXiv:2102.07920*, 2021.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *Advances in neural information processing systems*, 30, 2017.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. The emergence of spectral universality in deep networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 1924–1932. PMLR, 2018.

Haozhi Qi, Chong You, Xiaolong Wang, Yi Ma, and Jitendra Malik. Deep isometric learning for visual recognition. In *International Conference on Machine Learning*, 2020.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Technical Report*, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Andries Rosseau, Robert Müller, and Ann Nowé. Preserving plasticity in continual learning via dynamical isometry. In *International Conference on Machine Learning*, 2026. arXiv:2606.09762.

Tianyu Ruan, Fengzhuo Zhang, Shuche Wang, and Shihua Zhang. Muon learns more robust and transferable features than Adam. *arXiv preprint arXiv:2606.09658*, 2026.

Ali Saheb Pasand, Johan Obando-Ceron, Aaron Courville, Pouya Bashivan, and Pablo Samuel Castro. Stable deep reinforcement learning via isotropic Gaussian representations. In *International Conference on Machine Learning*, 2026. arXiv:2602.19373.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations*, 2014.

Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. In *International Conference on Learning Representations*, 2017.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level Atari with human-level efficiency. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 30365–30380. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/schwarzer23a.html.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019. URL https://arxiv.org/abs/1909.08053.

Sahil Singla, Surbhi Singla, and Soheil Feizi. Improved deterministic l2 robustness on CIFAR-10 and CIFAR-100. In *International Conference on Learning Representations*, 2022.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 32145–32168. PMLR, 2023.

Ghada Sokar, Johan Samir Obando Ceron, Aaron Courville, Hugo Larochelle, and Pablo Samuel Castro. Don't flatten, tokenize! unlocking the key to softmoe's efficacy in deep RL. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8oCrlOaYcc.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. DeepNet: Scaling transformers to 1,000 layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Kevin Wang, Ishaan Javali, Michał Bortkiewicz, Tomasz Trzciński, and Benjamin Eysenbach. 1000 layer networks for self-supervised RL: Scaling depth can enable new goal-reaching capabilities. In *Advances in Neural Information Processing Systems*, 2025.

Shuche Wang, Fengzhuo Zhang, Jiaxiang Li, Dirk Bergemann, and Zhuoran Yang. Why Muon outperforms Adam: A curvature perspective. *arXiv preprint arXiv:2606.04662*, 2026.

Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel S. Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, 2018.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs VI: Feature learning in infinite-depth neural networks. In *International Conference on Learning Representations*, 2024.

Yang Yue, Rui Lu, Bingyi Kang, Shiji Song, and Gao Huang. Understanding, predicting and better resolving Q-value divergence in offline-RL. In *Advances in Neural Information Processing Systems*, 2023.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder Alexander Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. Demonstrating MuJoCo playground. In *Robotics: Science and Systems*, 2025. arXiv:2502.08844.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, 2019.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations*, 2019.

APPENDIX

Just Stack It: Near-Isometric Layers for Deeper Reinforcement Learning

A Reinforcement learning background 16
B Further related work 17
C Proofs 18
D Optimizers 20
E Benchmarks 21
F Networks and baselines 22
G Hyperparameters 23
H Additional results 23
I Measurements of isometry 27
J Ablations and design alternatives 29
K Continual supervised learning 33
L Per-task results 36

A REINFORCEMENT LEARNING BACKGROUND

Value functions and bootstrapping. An agent interacts with a Markov decision process with states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition kernel $P(s' \mid s, a)$, reward function $r(s, a)$ and discount factor $\gamma \in [0, 1)$ (Sutton & Barto, 2018). A policy $\pi(a \mid s)$ is evaluated by its action-value function $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t \geq 0} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a]$, which satisfies the Bellman equation

$$Q^\pi(s, a) = r(s, a) + \gamma \, \mathbb{E}_{s' \sim P(\cdot|s,a),\, a' \sim \pi(\cdot|s')} \big[Q^\pi(s', a') \big],$$

and the optimal action-value function satisfies the same equation with the expectation over a' replaced by a maximum. Temporal-difference methods turn this fixed-point equation into a regression: the prediction $Q_\theta(s, a)$ is regressed onto a target computed from the network's own predictions at the next state (Equation 1), and the target is treated as a constant when the gradient is taken. Multi-step and λ-returns mix rewards over several steps with a bootstrapped value at the end, which trades the bias of the bootstrapped target against the variance of the observed rewards.

Non-stationarity. Two properties distinguish this regression from supervised learning. The targets depend on the parameters, so every update changes the targets of the next one, and the inputs are states visited by the current or recent policies, whose distribution changes as the policy improves. The network therefore solves a sequence of related regression problems. This is the setting in which networks lose plasticity (Nikishin et al., 2022; Lyle et al., 2023; Dohare et al., 2024), and in which, as Section 3 shows, deeper networks lose the conditioning of their tangent kernel.

Algorithms. *PQN* (Gallici et al., 2025) is a Q-learning algorithm that collects data from many parallel environments, regresses the network onto λ-returns computed on these fresh transitions, and uses neither a replay buffer nor a target network; LayerNorm in the published network keeps this on-policy regression stable. *SAC* (Haarnoja et al., 2018a;b) is an off-policy actor-critic method: two critics are regressed onto entropy-regularized targets computed with slowly updated copies of themselves on data from a replay buffer, and the actor maximizes the critics' value plus an entropy bonus. *TD3* (Fujimoto et al., 2018) uses a deterministic actor, two critics with a clipped double-Q target, target-policy smoothing and delayed actor updates. *PPO* (Schulman et al., 2017) is on-policy: a value network is regressed onto bootstrapped λ-returns, and the policy is updated with a clipped surrogate objective weighted by advantages estimated with that value network. *Hindsight relabeling* (Andrychowicz et al., 2017) makes goal-conditioned RL with sparse rewards tractable by replacing the goal of a stored transition with a state that was actually reached. In every case, the network whose depth we vary is regressed onto bootstrapped targets that change with every update.

B FURTHER RELATED WORK

Initialization and dynamical isometry. Variance-preserving initializations keep the mean squared singular value of each layer at one (Glorot & Bengio, 2010; He et al., 2015). Saxe et al. (2014) showed that deep linear networks with orthogonal weights learn at a speed independent of depth, and Pennington et al. (2017; 2018) used free probability to show that the whole spectrum of the input-output Jacobian, not only its mean, determines trainability, and that orthogonal weights with suitable activations achieve dynamical isometry where Gaussian weights cannot. Mean-field analyses identify the depth scales over which signals propagate (Schoenholz et al., 2017), and on this basis convolutional networks with ten thousand layers (Xiao et al., 2018) and deep networks without skip connections or normalization (Martens et al., 2021) were trained. These results control the spectrum at initialization, whereas a non-stationary objective requires control throughout training. Our layer bounds the spectrum throughout training as long as the constants c_k of Proposition 1 remain small, with a bound that is independent of depth and invariant to the scale of the weights.

Depth scaling of residual networks. Residual branches are commonly down-weighted with depth. GPT-2 and Megatron-LM scale the initialization of residual output projections by $1/\sqrt{2L}$ (Radford et al., 2019; Shoeybi et al., 2019), DeepNet rescales the branches to train Transformers with a thousand layers (Wang et al., 2024), and Depth-μP derives a $1/\sqrt{L}$ branch multiplier under which hyperparameters transfer across depth (Yang et al., 2024; Bordelon et al., 2024). CompleteP (Dey et al., 2025) shows that $1/L$ is the only multiplier under which every layer keeps learning nonlinear features as depth grows. Our multiplier is CompleteP's. We apply it to a normalized branch next to a fixed isometric skip connection, where it yields a bound on the conditioning of the hidden layers that is independent of depth and invariant to the scale of the weights (Proposition 1), and it is a fixed multiplier rather than an initialization scale, which training would change.

Exactly isometric layers. Norm-preserving activations such as GroupSort and MaxMin (Anil et al., 2019) or Householder activations (Singla et al., 2022), combined with orthogonal weights, make each layer an exact isometry at every input, and orthogonality constraints with shifted ReLUs were used to train deep networks without normalization (Qi et al., 2020). A layer that is an isometry at every input preserves the norm of every input and is 1-Lipschitz, so it cannot increase the distance between two inputs, and over many layers distinct inputs can merge (Appendix C.3; Appendix K). Isometry at every input is therefore not sufficient for a well-conditioned tangent kernel, which also depends on the rank of the features.

Loss of plasticity. Networks trained on changing objectives lose the ability to learn (Ash & Adams, 2020; Berariu et al., 2021; Dohare et al., 2024), in RL in particular (Igl et al., 2021; Nikishin et al., 2022; Lyle et al., 2022; Sokar et al., 2023; Abbas et al., 2023). Proposed methods include resetting parameters, regularizing toward the initial weights or bounding their spectra (Kumar et al., 2025; Lewandowski et al., 2025), normalizing and projecting the weights (Lyle et al., 2024), and categorical value outputs (Lyle et al., 2025). Lyle et al. (2023; 2025) find that no single statistic of the network explains the loss and point to a poorly conditioned tangent kernel as a common factor. Our analysis starts from the same object and derives from it two quantities that an architecture can control, the isometry of the hidden layers and the rank of the features. Rosseau et al. (2026) relate plasticity to the conditioning of the kernel and to layer-wise isometry, which they obtain with penalties and norm-preserving activations. Saheb Pasand et al. (2026) regularize the features toward an isotropic Gaussian distribution.

Scaling the width. BRO (Nauman et al., 2024) scales a residual critic with layer normalization in width, together with resets, weight decay and optimistic exploration, and BBF (Schwarzer et al., 2023) scales the width of a residual convolutional network for sample-efficient Atari, together with resets and other training changes. SimBa outperforms the BRO network at lower cost (Lee et al., 2025a), so we include SimBa and SimbaV2 as our main baselines in control tasks, and leave BRO out of our analysis. Finally, mixtures of experts are shown to improve the scaling of value networks (Obando-Ceron et al., 2024). However, Sokar et al. (2025) trace most of this improvement to the tokenization of the encoder output, and not the mixture-of-experts. Since a setting with tokenization is not the focus of our work, we leave this out of our analysis.

C PROOFS

C.1 THE SUM OF TWO ISOMETRIES

Let $\boldsymbol{S}$ and $\boldsymbol{Q}$ be orthogonal $n \times n$ matrices. We show that their sum is, in general, far from an isometry, whereas $\boldsymbol{S} + \varepsilon\boldsymbol{Q}$ is within ε of one.

Write $\boldsymbol{S} + \boldsymbol{Q} = \boldsymbol{S}(\boldsymbol{I} + \boldsymbol{O})$ with $\boldsymbol{O} = \boldsymbol{S}^\top\boldsymbol{Q}$. Multiplying by the orthogonal matrix $\boldsymbol{S}$ does not change singular values, so it suffices to look at $\boldsymbol{I} + \boldsymbol{O}$. The matrix $\boldsymbol{O}$ is orthogonal, hence normal, with eigenvalues $e^{i\vartheta_j}$ on the unit circle; the angles ϑ_j are the rotation angles of the relative rotation between $\boldsymbol{S}$ and $\boldsymbol{Q}$. Then $\boldsymbol{I} + \boldsymbol{O}$ is normal with eigenvalues $1 + e^{i\vartheta_j}$, and the singular values of a normal matrix are the moduli of its eigenvalues:

$$\sigma_j(\boldsymbol{S} + \boldsymbol{Q}) = \left|1 + e^{i\vartheta_j}\right| = 2\left|\cos(\vartheta_j/2)\right|. \tag{7}$$

For a generic pair of orthogonal matrices the angles ϑ_j spread over the whole circle (for a relative rotation drawn uniformly from the orthogonal group they become uniformly distributed as n grows), so the singular values fill the interval $[0, 2]$, with a density $2/(\pi\sqrt{4 - \sigma^2})$ that does not vanish at zero. The sum of two exact isometries is therefore arbitrarily ill conditioned.

Closeness to an isometry, on the other hand, is preserved under a small perturbation. By Weyl's inequality for singular values, $|\sigma_j(\boldsymbol{A} + \boldsymbol{E}) - \sigma_j(\boldsymbol{A})| \le \|\boldsymbol{E}\|_2$, so

$$\sigma_j(\boldsymbol{S} + \varepsilon\boldsymbol{Q}) \in [\,1 - \varepsilon,\ 1 + \varepsilon\,] \qquad \text{for every } j. \tag{8}$$

This is the observation behind our layer: an isometric path can be kept only if what is added to it is small.

C.2 PROOF OF PROPOSITION 1

Setting. Layer k maps $\boldsymbol{h}_{k-1}$ to $\boldsymbol{h}_k = \boldsymbol{S}_k\boldsymbol{h}_{k-1} + \frac{1}{K}\phi_{\boldsymbol{b}_k}(\nu(\boldsymbol{z}_k))$ with $\boldsymbol{z}_k = \boldsymbol{W}_k\boldsymbol{h}_{k-1}$. Write $r_k = \mathrm{rms}(\boldsymbol{z}_k)$ and $\boldsymbol{u}_k = \boldsymbol{z}_k/r_k$, so that $\|\boldsymbol{u}_k\|_2^2 = n$. We prove the three statements of the proposition in turn.

Step 1: each layer is within c_k/K of an isometry. At a point where the layer is differentiable, its Jacobian is

$$\boldsymbol{J}_k = \boldsymbol{S}_k + \frac{1}{K}\,\boldsymbol{D}_k\boldsymbol{N}_k\boldsymbol{W}_k, \tag{9}$$

where $\boldsymbol{D}_k$ is the diagonal $0/1$ matrix of the shifted ReLU (a one where $\boldsymbol{u}_k > \boldsymbol{b}_k$) and $\boldsymbol{N}_k = \partial\nu/\partial\boldsymbol{z}$ is the Jacobian of the normalization,

$$\boldsymbol{N}_k = \frac{1}{r_k}\Big(\boldsymbol{I} - \frac{\boldsymbol{u}_k\boldsymbol{u}_k^\top}{n}\Big). \tag{10}$$

The matrix in parentheses is an orthogonal projection, because $\|\boldsymbol{u}_k\|_2^2 = n$; hence $\|\boldsymbol{N}_k\|_2 = 1/r_k$. Together with $\|\boldsymbol{D}_k\|_2 \le 1$,

$$\Big\|\frac{1}{K}\boldsymbol{D}_k\boldsymbol{N}_k\boldsymbol{W}_k\Big\|_2 \;\le\; \frac{1}{K}\cdot\frac{\|\boldsymbol{W}_k\|_2}{r_k} \;=\; \frac{c_k}{K}. \tag{11}$$

Since $\boldsymbol{S}_k$ is orthogonal, all its singular values equal one, and Weyl's inequality places every singular value of $\boldsymbol{J}_k$ in $[1 - c_k/K,\ 1 + c_k/K]$.

Step 2: the product. The Jacobian of the hidden layers is $\boldsymbol{J}_K \cdots \boldsymbol{J}_1$. For square matrices the largest singular value of a product is at most the product of the largest singular values, and the smallest is at least the product of the smallest, so every singular value σ of the stack satisfies

$$\prod_{k=1}^{K}\Big(1 - \frac{c_k}{K}\Big) \;\le\; \sigma \;\le\; \prod_{k=1}^{K}\Big(1 + \frac{c_k}{K}\Big). \tag{12}$$

The upper bound follows from $1 + x \le e^x$: the product is at most $\exp(\sum_k c_k/K) = e^{\bar{c}}$. For the lower bound we use $\ln(1-x) \ge -x - x^2$ on $[0, 1/2]$, which is where the assumption $c_k \le K/2$ enters:

$$\prod_k \Big(1 - \frac{c_k}{K}\Big) \;\ge\; \exp\Big(-\sum_k \frac{c_k}{K} - \sum_k \frac{c_k^2}{K^2}\Big) \;\ge\; \exp\Big(-\bar{c}\,\big(1 + \max_k c_k/K\big)\Big). \tag{13}$$

This is Equation 6. The exponents are means over the layers, which is why the bound does not depend on K.

Step 3: what c_k measures. Replacing $\boldsymbol{W}_k$ by $\alpha \boldsymbol{W}_k$ multiplies both $\|\boldsymbol{W}_k\|_2$ and r_k by α, so c_k is invariant to the scale of the weights. Writing $c_k = \kappa_k / \mathrm{rms}(\boldsymbol{h}_{k-1})$ with

$$\kappa_k = \frac{\|\boldsymbol{W}_k\|_2 \, \|\boldsymbol{h}_{k-1}\|_2}{\|\boldsymbol{W}_k \boldsymbol{h}_{k-1}\|_2} \tag{14}$$

shows what remains: κ_k compares the largest gain of $\boldsymbol{W}_k$ with its gain on the current hidden state. It equals one when $\boldsymbol{W}_k$ is a multiple of an orthogonal matrix, and it is at most the condition number of $\boldsymbol{W}_k$. A poorly conditioned weight matrix increases c_k only if the hidden state occupies its weak directions.

Step 4: the forward pass. For a threshold $b \le 0$ we have $|\max(u, b)| \le |u|$ for every real u, so $\|\phi_{\boldsymbol{b}_k}(\boldsymbol{u}_k)\|_2 \le \|\boldsymbol{u}_k\|_2 = \sqrt{n}$. The skip preserves the norm, hence $\|\boldsymbol{h}_k\|_2 \le \|\boldsymbol{h}_{k-1}\|_2 + \sqrt{n}/K$; summing over the K layers and dividing by $\sqrt{n}$ gives $\mathrm{rms}(\boldsymbol{h}_K) \le \mathrm{rms}(\boldsymbol{h}_0) + 1$. □

Remarks. In practice $\nu(\boldsymbol{z}) = \boldsymbol{z}/\sqrt{\mathrm{rms}(\boldsymbol{z})^2 + \epsilon}$ with a small ϵ; then $\|\boldsymbol{N}_k\|_2 \le 1/r_k$ and $\|\boldsymbol{u}_k\|_2 \le \sqrt{n}$, and every step holds unchanged. With a bias, $\boldsymbol{z}_k = \boldsymbol{W}_k \boldsymbol{h}_{k-1} + \boldsymbol{a}_k$, the proof goes through with $c_k = \|\boldsymbol{W}_k\|_2 / \mathrm{rms}(\boldsymbol{z}_k)$, which is invariant to a joint rescaling of $\boldsymbol{W}_k$ and $\boldsymbol{a}_k$. The bound of Step 1 is nearly tight: on Atari, the measured deviation $\|\boldsymbol{J}_k - \boldsymbol{S}_k\|_2$ of a layer reaches on average 92% of c_k/K under Muon (Appendix I).

C.3 Distances between hidden states

The same reasoning bounds how the hidden layers change the distance between any two hidden states. Let the branch $g_k = \frac{1}{K}\,\phi_{\boldsymbol{b}_k} \circ \nu \circ \boldsymbol{W}_k$ be L_k-Lipschitz with $L_k < 1$ on a convex set containing the hidden states of interest; by Step 1, $L_k \le \sup_{\boldsymbol{h}} c_k(\boldsymbol{h})/K$. This condition is stronger than the assumption of Proposition 1, which constrains c_k only at the hidden states themselves. The skip preserves distances and the branch changes them by at most $L_k \|\boldsymbol{h} - \boldsymbol{h}'\|_2$, so

$$(1 - L_k)\,\|\boldsymbol{h} - \boldsymbol{h}'\|_2 \;\le\; \|F_k(\boldsymbol{h}) - F_k(\boldsymbol{h}')\|_2 \;\le\; (1 + L_k)\,\|\boldsymbol{h} - \boldsymbol{h}'\|_2, \tag{15}$$

as in invertible residual networks (Behrmann et al., 2019). Over K layers the distance between any two hidden states changes by a factor between $\prod_k (1 - L_k)$ and $\prod_k (1 + L_k)$, which obey Equation 6 with c_k replaced by its supremum. In particular, the map computed by the hidden layers is injective: it cannot map two distinct inputs to the same representation. Since the sum of the squared pairwise distances of n points equals $2n$ times their total variance, the squares of the same factors also bound the change in the total variance of a set of hidden states, so the representations cannot collapse. How this variance is distributed over directions, and hence the rank of the features, is not fixed by these bounds; we measure it and find it high in all our networks (Section 5.2).

A network of orthogonal matrices and norm-preserving activations (Anil et al., 2019) can, although its Jacobian is orthogonal at almost every input. Such a network is continuous and piecewise linear with an orthogonal matrix on every piece; two orthogonal matrices that agree on the hyperplane between adjacent pieces differ by the reflection across it, so the network is nonlinear only through reflections of its input space across these hyperplanes, and such a reflection can map two inputs that are mirror images of each other to the same point. An exactly isometric Jacobian preserves small perturbations around each input, but not the distances between inputs, on which the forward factor of Equation 4 depends.

D OPTIMIZERS

D.1 THE SECOND-ORDER VIEW

To second order, a preconditioned step $\Delta\theta = -\eta\, \boldsymbol{P}\boldsymbol{J}_\theta^\top \boldsymbol{r}$ on the squared loss with residuals $\boldsymbol{r}$ changes the loss by

$$-\eta\, \boldsymbol{r}^\top \boldsymbol{J}_\theta \boldsymbol{P}\boldsymbol{J}_\theta^\top \boldsymbol{r} + \tfrac{\eta^2}{2}\, \boldsymbol{r}^\top \boldsymbol{J}_\theta \boldsymbol{P}\mathbf{H}\boldsymbol{P}\boldsymbol{J}_\theta^\top \boldsymbol{r}, \qquad \mathbf{H} = \boldsymbol{J}_\theta^\top \boldsymbol{J}_\theta + \textstyle\sum_i r_i \nabla_\theta^2 f_i. \tag{16}$$

The Gauss-Newton part $\boldsymbol{J}_\theta^\top \boldsymbol{J}_\theta$ has the nonzero spectrum of the kernel, and the preconditioned curvature $\boldsymbol{P}^{1/2}\boldsymbol{J}_\theta^\top \boldsymbol{J}_\theta \boldsymbol{P}^{1/2}$ has the same nonzero eigenvalues as $\boldsymbol{J}_\theta \boldsymbol{P}\boldsymbol{J}_\theta^\top$: what the natural gradient and its Kronecker-factored approximations invert is the conditioning of the kernel. The remaining term, the curvature of the network weighted by the residuals, is inverted by none of them. In our layer each branch enters the output through the $1/K$ multiplier and a normalization of unit scale, so the second derivative of the output with respect to the parameters of one layer is of order $1/K$, and the sum over K layers stays of order one. Dey et al. (2025) use the same argument to show that $1/K$ keeps every order of a block's expansion in play as depth grows.

D.2 ADAM AND RADAM

RAdam (Liu et al., 2020) is Adam (Kingma & Ba, 2015) with a correction for the variance of the adaptive learning rate in the first updates, when the estimate of the second moment of the gradient rests on few samples. The correction factor approaches one as training proceeds, after which the two optimizers take the same steps. We keep the first-order optimizer of each published agent, RAdam for PQN and Adam for SAC, TD3, PPO and SimbaV2, so the two differ only early in training, and we expect the choice between them to matter little for our results.

D.3 MUON

Muon (Jordan et al., 2024) keeps a momentum $\boldsymbol{M}$ of the gradient of each weight matrix and steps along an approximation of its orthogonal polar factor: if $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ is a compact singular value decomposition, the update is $-\eta\, \boldsymbol{U}\boldsymbol{V}^\top$. It is computed with five steps of the Newton-Schulz iteration $\boldsymbol{X} \leftarrow a\boldsymbol{X} + b\,\boldsymbol{X}\boldsymbol{X}^\top\boldsymbol{X} + c\,(\boldsymbol{X}\boldsymbol{X}^\top)^2\boldsymbol{X}$, which moves all singular values of $\boldsymbol{X}$ close to one. Three properties of this update relate to our analysis.

First, among all updates of spectral norm at most η, $-\eta\, \boldsymbol{U}\boldsymbol{V}^\top$ decreases the linearized loss the most, so Muon is steepest descent under the spectral norm (Bernstein & Newhouse, 2024). Since $\|\Delta\boldsymbol{W}\boldsymbol{h}\|_2 \le \|\Delta\boldsymbol{W}\|_2\, \|\boldsymbol{h}\|_2$, one step changes the output of a layer by at most η times the norm of its input, for every input, and by Weyl's inequality it moves every singular value of the weight matrix by at most η.

Second, the update has equal singular values along all directions of its gradient, as the Jacobians of an isometric network have equal singular values along all directions of the hidden states. Gradients of weight matrices, in contrast, are typically dominated by a few singular directions (Jordan et al., 2024), and a gradient step changes the layer mostly along these. Wang et al. (2026) show on quadratic models that the orthogonalized update spreads its energy evenly over the directions of curvature, whereas gradient descent concentrates it on the directions of highest curvature, and they measure a lower curvature along the updates of Muon than along those of Adam in language models.

Third, the Newton-Schulz iterate is $\boldsymbol{M}$ multiplied from the right by a polynomial in $\boldsymbol{M}^\top\boldsymbol{M}$, so its rows lie in the span of the rows of $\boldsymbol{M}$. For a linear layer the gradient is $\sum_i \boldsymbol{b}_i\boldsymbol{h}_i^\top$, with $\boldsymbol{h}_i$ the inputs of the layer and $\boldsymbol{b}_i$ the backward signals, so a Muon update, like a gradient step, acts only on the span of the inputs that entered the momentum. Adam normalizes each coordinate of the update separately: for a single input, its first step is close to $\operatorname{sign}(\boldsymbol{b})\operatorname{sign}(\boldsymbol{h})^\top$, whose rows are not in the span of $\boldsymbol{h}$. Components of the weights along directions that the hidden states do not occupy do not change the output on the data, so the loss does not act on them, and without weight decay they remain. This is one way in which $\|\boldsymbol{W}_k\|_2$, and with it the constant c_k of Proposition 1, can grow while the gain of $\boldsymbol{W}_k$ on the hidden states does not.

Two findings in language and vision models match what we observe in RL. Liu et al. (2025) measure the entropy of the normalized squared singular values of every weight matrix during pre-training and

find it higher under Muon than under AdamW at every checkpoint and for more than 90% of the matrices, which they attribute to updates along more diverse directions. Ruan et al. (2026) find hidden features of higher effective rank under Muon than under Adam or SGD, and prove for a one-layer classifier that Muon reaches a given loss with features of higher effective rank. Neither concerns non-stationary objectives. On Atari, Muon keeps our weight matrices close to orthogonal: their median condition number at the end of training is 6.1, against 2,600 under RAdam and 3,069 under Kron, a stricter measure than the entropy of the spectrum, which is set by its bulk rather than its smallest singular value. The benefit of Muon does not come from orthogonal weights alone, however, since an orthogonality penalty with a first-order optimizer keeps the weights orthogonal but does not reach the score of Muon (Appendix J).

D.4 KRON

Kron, the Kronecker-factored preconditioner of PSGD (Li, 2018), maintains for each weight matrix two factors, one acting on the inputs of the layer and one on its backward signals, and fits them online so that the preconditioned gradients have whitened statistics. It targets the same Kronecker-factored curvature as K-FAC and Shampoo (Martens & Grosse, 2015; Gupta et al., 2018) without explicit matrix inverses. Muon can be read as Shampoo without the accumulation of the factors over steps (Jordan et al., 2024), and Che (2026) combine the two, computing the polar factor of Muon in coordinates whitened by Kronecker factors. At depth 32 on Atari, Muon increases the time per run over RAdam by about a third and Kron by about a half (Appendix H.6).

E BENCHMARKS

Atari-10. Ten games of the Arcade Learning Environment (Bellemare et al., 2013), selected by Aitchison et al. (2023) so that their scores predict those on the full suite:

Amidar, Battle Zone, Bowling, Double Dunk, Frostbite, Kung-Fu Master,
Name This Game, Phoenix, Q*bert and River Raid.

Agents train for 40M frames. PQN is scored with its standard evaluation: the greedy policy on eight test environments, averaged over the last 10% of evaluations. PPO is scored on its training episodes over the last 10% of training, as is standard for PPO. Scores are human-normalized, $(G - G_{\text{random}})/(G_{\text{human}} - G_{\text{random}})$, with the reference scores of Table 2.

DeepMind Control. Tasks of the DeepMind Control suite (Tassa et al., 2018) in MuJoCo Playground (Zakka et al., 2025), at the budgets of the published agents. For the main results we use the tasks on which the published agents leave room for improvement, because depth can only show its effect where the shallow network has not already solved the task. SAC and TD3 use the eight tasks on which the published SAC agent does not reach its maximum return:

AcrobotSwingup, FingerTurnHard, FishSwim, HopperHop,
HopperStand, HumanoidRun, SwimmerSwimmer6 and WalkerRun.

Three further tasks with sparse rewards, PendulumSwingup, CartpoleSwingupSparse and AcrobotSwingupSparse, are left out of the SAC set because every agent, the published one included, succeeds or fails on them by chance from seed to seed. PPO uses the sixteen tasks on which the published PPO agent does not reach its maximum return (Table 22). We also ran the remaining tasks of the suite, except the two Reacher tasks, which do not run with the simulator version we use. On these tasks the published agents are already close to their maximum return, so the results say little about depth (Table 23). Each run is scored by its final evaluation return divided by the published return of the same algorithm on the same task; a run that diverges scores zero.

Locomotion. Eleven Playground locomotion tasks with SAC for 10M steps (Table 21): all locomotion tasks except those of the three heaviest humanoid robots, Apollo, Berkeley Humanoid and G1, which cost three to seven times as much as the others to simulate, and Barkour, which does not fit in the memory of one accelerator. The eleven tasks cover every other robot family, Go1, H1, Op3 and Spot; the two tasks of the T1 humanoid are reported separately (Appendix H). Since no

Table 2: **Atari-10 reference scores** used for human normalization.

Game	Random	Human
Amidar	5.8	1,719.5
Battle Zone	2,360.0	37,187.5
Bowling	23.1	160.7
Double Dunk	−18.6	−16.4
Frostbite	65.2	4,334.7
Kung-Fu Master	258.5	22,736.3
Name This Game	2,292.3	8,049.0
Phoenix	761.4	7,242.6
Q*bert	163.9	13,455.0
River Raid	1,338.5	17,118.0

SAC returns are published for these tasks, each run is scored by the mean of its last three evaluation returns divided by the same quantity for the published SAC agent, which we ran at its published setting with two seeds.

Craftax and goal-conditioned RL. Craftax-Symbolic (Matthews et al., 2024) with PQN for 100M steps, scored by the episode return (at most 226) over the last 10% of training. The goal-conditioned tasks of JaxGCRL (Bortkiewicz et al., 2025) with SAC and hindsight relabeling, in the data regime and at the budgets of Wang et al. (2025): all tasks of their study except the two humanoid mazes (Humanoid U-Maze and Humanoid Big Maze), the most expensive to simulate, scored by the number of steps per episode spent within the goal radius (out of 1,001), averaged over the last five evaluations.

Scope. The experiments span five algorithms (PQN, SAC, TD3, PPO and SAC with hindsight relabeling), five benchmark suites and a controlled supervised task, depths from 1 to 64, three optimizers, and more than a dozen ablations and controls. Apart from the selections stated above, tasks were left out only for reasons of compute.

Aggregation. We report the interquartile mean over all runs of all tasks, with 95% stratified bootstrap confidence intervals in which runs are resampled within each task (Agarwal et al., 2021).

F Networks and baselines

We compare networks at equal depth, the number of weight matrices between the input layer and the output layer. At equal depth, all networks also have the same number of parameters in these matrices (Table 3). The input and output layers depend on the task and are the same for all networks, except that SimbaV2 and SimBa keep their own input layer of width 256.

Ours. A linear input layer to width 512 followed by the normalization ν, K of our layers (Equation 5) at width 512, and a linear output layer. In SAC and TD3 our layers form the critic, in PPO on DeepMind Control the value network, in PPO on Atari the value head behind the shared encoder, and in PQN the layers behind the convolutional encoder of Mnih et al. (2015). Policy networks are those of the published agents. Using our layers in the policy network as well did not improve PPO in single-seed checks: a policy of 16 of our layers at width 128 scores 1.31 against 1.48 on DeepMind Control, and 1.02 against 1.03 on locomotion.

Baseline. The published network of each agent: one hidden layer of 512 units behind the convolutional encoder in PQN (Gallici et al., 2025), two hidden layers of 256 units in the SAC critic (Zakka et al., 2025), and the networks of CleanRL (Huang et al., 2022) and Playground in PPO.

Plain. The published network with more hidden layers of width 512. A plain network with L hidden layers has depth $L - 1$, since its first hidden layer is its input layer.

Multi-skip. The architecture of Castanyer et al. (2025): blocks of two weight matrices at width 512, with additional skip connections that add the encoder output after every block, and their initial-

Table 3: **Networks at equal depth.** Depth is the number of weight matrices between the input and the output layer; parameters are those of these matrices. PPO on Atari also uses our network at depth 1.

Depth	Parameters (M)	Ours (layers)	SimbaV2, SimBa (blocks)	Multi-skip (blocks)	Plain (hidden layers)
2	0.52	2	1	1	3
4	1.05	4	2	2	5
6	1.57	6	3	3	7
8	2.10	8	4	4	9
10	2.62	10	5	5	11
16	4.19	16	8	8	17
32	8.39	32	16	16	33

ization. It is trained with Kron (Li, 2018) at the learning rate prescribed by Castanyer et al. (2025) (Appendix G), and its blocks are stacked in the same trainers as ours.

SimbaV2 and SimBa. The residual blocks released by Lee et al. (2025b) and Lee et al. (2025a), at width 256, so that $K/2$ blocks precisely match K of our layers in depth and parameters: each block has two weight matrices ($256 \to 1024 \to 256$) and $8 \cdot 256^2 = 2 \cdot 512^2$ parameters. SimbaV2 runs at its published setting, as tuned by its authors, SimBa at the best learning rate of our sweep.

Contrastive RL. The scaled contrastive agent of Wang et al. (2025) at depths 4, 16 and 64. We use their published results on the same tasks.

G Hyperparameters

Tables 4 and 5 the full configurations. Our network uses one setting per algorithm and benchmark at every depth, chosen at depth 32. The $1/K$ branch multiplier of our layer is the one of CompleteP (Dey et al., 2025), under which the best learning rate transfers across depth, and the following checks show that this transfer holds approximately in RL, so that the choice of setting does not produce the gains with depth. With SAC on DeepMind Control (three seeds), the better of the two best settings scores 1.70, 2.00, 2.02, 2.12 and 2.24 at depths 2, 4, 8, 16 and 32, against 1.55, 2.00, 2.02, 2.12 and 2.24 for the setting chosen at depth 32: tuning at each depth raises only the score of the shallowest network, which remains the lowest. With PQN under Kron, the rate we use is the best of four at depth 2 and within 3% of the best at depth 32 (four games, two seeds). With PPO on DeepMind Control, half the rate scores 1.31 instead of 1.26 at depth 8, against 1.48 at depth 32 (one seed). We therefore use one setting at every depth, so that depths and networks are compared at the same setting. On Playground, the learning rate of the parameters inside the residual branches is multiplied by a constant, the *branch learning-rate factor*.

H Additional results

Table 6 gives the scores at depth 32 with their confidence intervals, next to the baseline and the prior deep network at its best depth.

H.1 SAC on DeepMind Control

Table 7 gives our network on the eight DeepMind Control tasks under three optimizers and with TD3, next to the networks it is compared with. Performance rises with depth under every optimizer, SimbaV2 and SimBa stay below our network at every depth, and the plain critic collapses completely, with or without the training improvements of SimbaV2. With TD3, for which Playground publishes no setting, our critic doubles the score of the published critic at depth 32; four of its HumanoidRun runs diverged and count as zero.

At 8.4M parameters, depth 32 scores highest, depth 8 at twice the width slightly lower and depth 2 at four times the width lowest; at 2.1M parameters, depth 8 scores above depth 2 at twice the width

Table 4: Our network uses the same setting at every depth, chosen at depth 32 from the sweep listed. All other settings are those of the published agents.

PQN, Atari-10	
Budget	40M frames, three seeds, constant learning rate as published
Our network	RAdam or Muon at $2.5 \cdot 10^{-4}$, Kron at $8.3 \cdot 10^{-5}$
Sweep at depth 32	Muon $6.3 \cdot 10^{-5}$ to $2 \cdot 10^{-3}$; Kron $4.2 \cdot 10^{-5}$ to $3.3 \cdot 10^{-4}$
Exploration	final exploration rate 0.005 for every network, as in Castanyer et al. (2025)
Baselines	Baseline and plain network: RAdam at $2.5 \cdot 10^{-4}$, as published; multi-skip: Kron at $8.3 \cdot 10^{-5}$, one third of the published Adam rate, as prescribed by Castanyer et al. (2025)
PPO, Atari-10	
Budget	40M frames, three seeds, learning rate annealed to zero as published
Our network	Kron at $1.7 \cdot 10^{-4}$ on the whole agent
Sweep at depth 32 (one seed)	Kron $4.2 \cdot 10^{-5}$ to $3.3 \cdot 10^{-4}$; Muon $2.5 \cdot 10^{-4}$ to 10^{-3}; Adam $1.3 \cdot 10^{-4}$ to $2 \cdot 10^{-3}$
Baselines	Baseline: Adam at $2.5 \cdot 10^{-4}$, as published (Huang et al., 2022); multi-skip: Kron at $8.3 \cdot 10^{-5}$ with the settings of its released code
PQN, Craftax-Symbolic	
Budget	100M steps, three seeds, learning rate decayed linearly to zero
Our network	width 1024, RAdam at 10^{-3}
Baseline	the published network (four layers of 1024 units with LayerNorm) at 10^{-4}, as published (also tried at $3 \cdot 10^{-4}$ and 10^{-3})

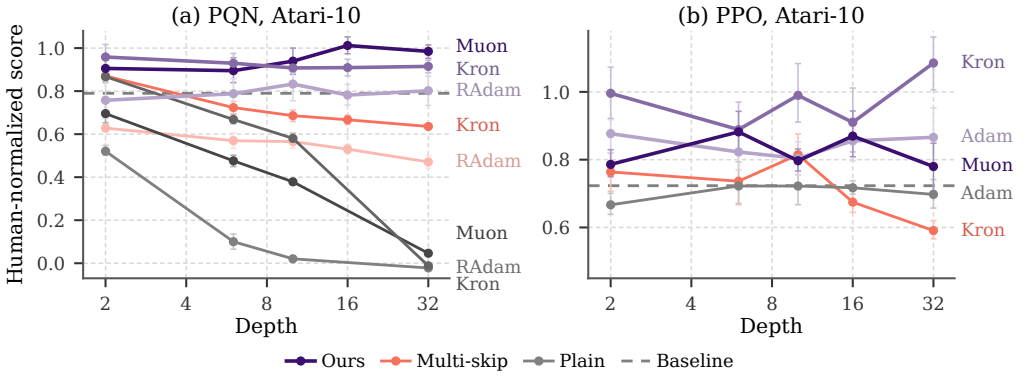

Figure 4: **Atari-10 against depth under every optimizer**, the panels (d) and (e) of Figure 2 with all networks and optimizers we ran. IQM with 95% confidence intervals; labels give the optimizer. Dashed: the published network under its published optimizer. Grey: the plain network, in (a) under each optimizer and in (b) with the published settings, where plain layers form the value head. In (a), 9 of the 30 plain runs under Kron at depth 32 end with non-finite values during training and count as they are.

(Table 8). Wider SimbaV2 networks, with 8 and 16 blocks at width 512 and 16.8M and 33.6M parameters, score 1.98 and 1.86.

H.2 The T1 humanoid

On the two tasks of the T1 humanoid (three seeds), our critic scores above the Baseline at every depth (Table 9).

H.3 Craftax-Symbolic

With PQN on Craftax-Symbolic, the return of our network rises with depth and is above that of the published network at every depth (Table 10). At depth 32, our agent reaches the rarer achievements,

Table 5: MuJoCo Playground and JaxGCRL. As Table 4.

SAC and TD3, MuJoCo Playground	
Budget	5M or 10M steps on DeepMind Control, as published; 10M steps on locomotion; three seeds
Our critic	Adam at 10^{-3}, decayed linearly to $5 \cdot 10^{-4}$; branch learning-rate factor 4 on DeepMind Control and 2 on locomotion
Training improvements	categorical critic, return-based reward scaling and initial temperature 0.01, taken from SimbaV2 (Lee et al., 2025b); used by our critic and by the Baseline
Sweep at depth 32	Adam 10^{-3} and $2 \cdot 10^{-3}$ with factors 2, 4 and 8 (one seed), the two best at every depth (three seeds); on locomotion $3 \cdot 10^{-4}$ to $2 \cdot 10^{-3}$ with factors 2 and 4, at depths 8 and 32; Muon $5 \cdot 10^{-4}$ to $2 \cdot 10^{-3}$ and Kron $5 \cdot 10^{-5}$ to $5 \cdot 10^{-4}$ for Table 7
TD3	Adam at $5 \cdot 10^{-4}$, branch factor 2; swept from 0.35 to 2 times this rate with factors 2 and 4 at depth 32 (one or two seeds)
Baselines	SimbaV2 as published (Adam at 10^{-4}, decayed to half); SimBa at Adam 10^{-3} without weight decay, the best of a sweep; the published critic at 10^{-3}, with and without the training improvements; for TD3, the critic of the published SAC configuration, since no TD3 setting is published
PPO, DeepMind Control	
Budget	60M or 100M steps, as published; three seeds; constant learning rate
Our value network	Adam at 10^{-3}, branch factor 8
Other changes	policy network as published but trained with Muon at 10^{-3}; quantile value head and return scaling, also given to the Baseline
Sweep at depth 32 (one seed)	learning rate $\times 0.5$, $\times 1$, $\times 2$; branch factor 0, 2, 8, 32; other assignments of optimizers to the two networks
Baselines	Baseline: Adam at 10^{-3}, with our value head; multi-skip: Kron at $3.3 \cdot 10^{-4}$ in the policy and value networks
SAC with hindsight relabeling, JaxGCRL	
Budget	50M to 200M steps, as in Wang et al. (2025); three seeds unless stated
Our critic	our layers in the critic only; the data regime and learning rate of Wang et al. (2025) (Adam at $3 \cdot 10^{-4}$, batch 512, 512 environments), no branch factor, no tuning
Baseline	the repository's critic in the same regime

such as entering the dungeon and defeating an orc mage, far more often than the published network, which is ahead on two early achievements, making a stone pickaxe and eating a cow.

H.4 Goal-conditioned RL

Figure 8 repeats Figure 3 with the range over seeds, Figure 7 gives the final values relative to contrastive RL, and Table 11 lists them.

H.5 Other settings

Behind a deeper encoder. Behind the Impala encoder (Espeholt et al., 2018) on Atari-10 with PQN (three seeds; Figure 9), the published network scores 1.12 and the multi-skip network at depth 16 scores 1.04. Our network at depth 16 scores 1.17 under RAdam, 1.23 under Muon and 1.39 under Kron.

PPO on locomotion. With PPO on the Playground locomotion tasks (one seed per task; Figure 11), a value network of our layers scores 1.02 to 1.04 times the published agent at every depth from 2 to 32 at the published budgets, 0.95 to 1.01 at 10M environment steps, and between 0.88 and 1.03 at every learning rate and branch factor we ran at depth 32. The published agent leaves little room for improvement on these tasks.

Table 6: **Results at depth 32.** IQM with stratified bootstrap 95% confidence intervals. Scores: human-normalized (Atari), return relative to the published agent (DeepMind Control, Locomotion; for TD3 the published SAC agent), return (Craftax), relative time at goal (JaxGCRL). The prior deep network is shown at its best depth; at every depth it has as many parameters as ours at that depth. Our hidden layers have 8.4M parameters (Craftax 33.6M, JaxGCRL 4.2M). Bold: interval disjoint from the others.

Benchmark	Algorithm	Baseline	Prior deep network (depth)		Ours, depth 32
Atari-10	PQN[a]	0.90 [0.87, 0.93]	Multi-skip (2)	0.87 [0.84, 0.90]	**0.98**[c] [0.95, 1.01]
Atari-10	PPO[b]	0.81 [0.76, 0.86]	Multi-skip (10)	0.81 [0.76, 0.88]	**1.09** [1.01, 1.16]
DM Control	PPO	1.08[d] [1.02, 1.16]	Multi-skip (4)	0.83 [0.79, 0.87]	1.28[c] [1.14, 1.43]
DM Control	SAC	1.51[d] [1.32, 1.80]	SimbaV2 (16)	1.92 [1.85, 1.99]	**2.24** [2.16, 2.38]
Locomotion	SAC	1.37[d] [1.31, 1.45]	SimbaV2 (32)	2.03 [1.95, 2.11]	**2.34** [2.27, 2.42]
DM Control	TD3	1.00 [0.93, 1.08]	–	–	**2.05** [1.78, 2.41]
Craftax	PQN[e]	17.0	–	–	23.2
JaxGCRL	SAC+HER[f]	0.38	Contrastive RL	1.00	1.42

[a] Ours and the baseline under Muon, the multi-skip network under Kron; the baseline under its published RAdam: 0.79. [b] All under Kron; the baseline under its published Adam: 0.72. [c] At depth 16: 1.01 (PQN) and 1.36 (PPO). [d] With the training improvements of SimbaV2 (SAC) or our value network (PPO). [e] Mean of three runs; ours at depth 1: 17.6. [f] Time at goal relative to contrastive RL at its best depth, median over eight tasks; baseline (their SAC with hindsight relabeling) and contrastive RL from the published results of Wang et al. (2025); ours at depth 64.

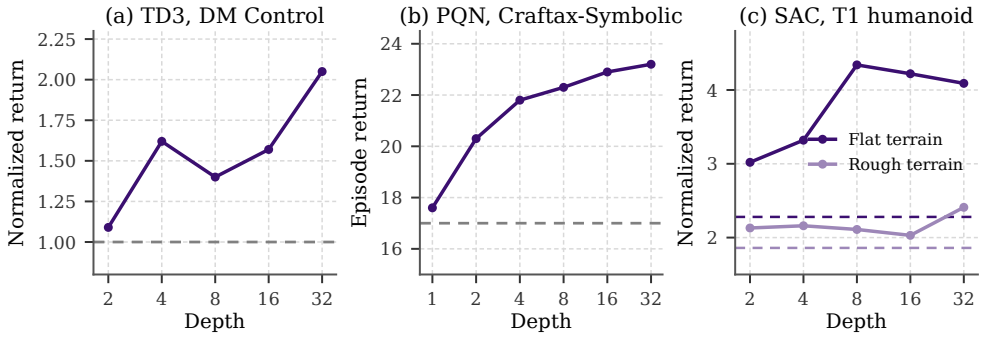

Figure 5: **Further algorithms and benchmarks against depth. (a)** TD3 with our critic on the eight DeepMind Control tasks, IQM of returns relative to the published SAC agent; dashed: the published critic trained with TD3. **(b)** PQN with our network on Craftax-Symbolic, episode return (mean of three runs); dashed: the published network. **(c)** SAC with our critic on the two T1 humanoid tasks, returns relative to the published SAC agent (mean over three seeds); dashed: the baseline of each task.

Learning speed and variability across seeds. At depth 32 on Atari, 26 of our 30 runs reach the final score of the published network under Muon and under Kron, after a median of 53% and 48% of training, against 17 of the 30 multi-skip runs, after 74% (Figure 10a). With our network under Muon, the standard deviation of the score across seeds, averaged over the ten games, falls from 0.17 at depth 2 to 0.066 at depth 32 (Figure 10b).

H.6 Training time

Table 12 gives the wall-clock time of one training run against depth on one H200 GPU. On Atari the encoder and the environment dominate, so depth is cheap: at depth 32 our network costs 1.5 to 2.0 times the published network, and the multi-skip network 2.5 times. With SAC the critic updates dominate and the time grows linearly with depth, equally for our network and for SimbaV2 at equal parameter count.

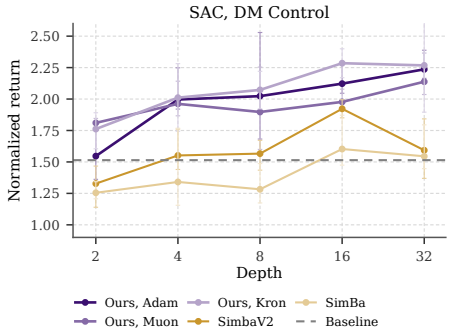

Figure 6: **SAC on the eight DeepMind Control tasks under three optimizers**: IQM of returns relative to the published SAC agent with 95% confidence intervals (Table 7). Dashed: the baseline, the published critic with the training improvements of SimbaV2.

Table 7: **SAC on the eight DeepMind Control tasks, by depth**: IQM of returns normalized by the published SAC agent. The Baseline scores 1.51; the published critic trained with TD3 scores 1.00. The plain rows use two seeds (three at depth 2).

Depth	2	4	8	16	32
Ours, Adam	1.55	2.00	2.02	2.12	2.24
Ours, Muon	1.81	1.96	1.90	1.98	2.14
Ours, Kron	1.76	2.01	2.07	2.28	2.27
SimbaV2	1.33	1.55	1.57	1.92	1.59
SimBa	1.25	1.34	1.28	1.60	1.54
Plain	1.49	1.65	1.42	0.53	0.06
Plain, with the training improvements	–	1.56	1.81	0.36	0.05
Ours, TD3	1.09	1.62	1.40	1.57	2.05

I MEASUREMENTS OF ISOMETRY

Runs. The measurements of Sections 3.1 and 5.2 use the runs of Section 5.1 repeated with checkpoints, 1,565 runs in total.

- *Atari-10, PQN*, ten games, three seeds, nine checkpoints per run. Our network at depths 2, 10 and 32 and its seven depth-32 variants (Table 14) under Muon, and at depth 32 with its variants under RAdam and Kron; the plain network under RAdam at depths 2, 6, 10 and 32 and under Muon at depths 2 and 10; the multi-skip network at depth 32 under all three optimizers; and, on Qbert and Phoenix, the plain network with normalize-and-project or spectral regularization (Table 15).
- *DeepMind Control, SAC*, the eight tasks of Section 5.1, three seeds: our critic at depths 2, 8 and 32, SimbaV2 at the same depths, the plain critic at depths 8 and 32, and the published critic.
- *DeepMind Control, PPO*, the sixteen tasks, three seeds: our value network at depths 2, 8 and 32, the multi-skip network at depths 2 and 32, and the published network.

The runs with checkpoints use the same settings as the scored runs.

Quantities. At the final checkpoint, on a fixed batch of states (128 states fixed at the first update on Atari, 512 states of the run's final sample on Playground), we compute: the singular values of the Jacobian of the hidden layers, from the input of the first to the output of the last, with values below 10^{-3} counted as lost, and from them the distance from isometry, the share within a factor of two of one, the fraction of lost directions, the mean squared singular value, the effective rank and the condition number; the rank of the features that reach the output layer, the number of their singular values that hold 99% of the sum (Kumar et al., 2021); the fraction of dormant units (Sokar et al., 2023); the weight norms; the gradient norm per hidden layer, of the temporal-difference loss on Atari and of the output on Playground; the top eigenvalue and the trace of the Hessian of the loss

Table 8: **Depth against width at equal parameters**, SAC on the eight DeepMind Control tasks. The wide versions of our network ran with two seeds, at the better of two learning rates on the first seed.

Network	Parameters (M)	IQM
Ours, depth 32, width 512	8.4	2.24
Ours, depth 8, width 1024	8.4	2.12
Ours, depth 2, width 2048	8.4	1.72
SimbaV2, 4 blocks, width 512	8.4	1.93
Ours, depth 8, width 512	2.1	2.02
Ours, depth 2, width 1024	2.1	1.67
SimbaV2, 1 block, width 512	2.1	1.76

Table 9: **The T1 humanoid**, SAC, three seeds: returns divided by those of the published SAC agent at its published setting, mean over seeds.

		Ours, depth				
Task	Baseline	2	4	8	16	32
T1JoystickFlatTerrain	2.28	3.02	3.32	4.34	4.22	4.09
T1JoystickRoughTerrain	1.86	2.13	2.16	2.11	2.03	2.41

(Atari); and the tangent kernel of the hidden layers, $\boldsymbol{K} = \boldsymbol{G}\boldsymbol{G}^\top$, whose rows are the gradients of the network's output on 128 probe states with respect to the parameters of the hidden layers. We restrict the kernel to the hidden layers because the kernel of the whole network is dominated by the convolutional encoder, which carries more than 99% of its trace on our networks, and because under an adaptive optimizer the kernel that governs learning is $\boldsymbol{J}\boldsymbol{P}\boldsymbol{J}^\top$, which rescales exactly these contributions.

Robustness. The share of singular values within a factor c of one correlates with the score, within tasks, at $+0.58$ to $+0.63$ on Atari, $+0.27$ to $+0.32$ with PPO and $+0.27$ to $+0.30$ with SAC for c between 1.5 and 4, and a smooth version, $\frac{1}{n}\sum_i \exp(-(\ln \sigma_i)^2/2s^2)$ for s between $\ln 1.5$ and $\ln 3$, gives $+0.59$ to $+0.60$, $+0.33$ and $+0.28$ to $+0.30$ (Table 1). The logarithmic condition number of the kernel correlates at -0.42, -0.30 and -0.49 (Atari, PPO, SAC), but reaches the numerical floor in most Atari runs, whose probe batches contain repeated states.

Frozen skips and competing explanations. Table 14 gives the frozen-skip manipulation of Section 5.2 under all three optimizers. Figure 14 tests the variables that prior work holds responsible on the same seven networks (score = IQM of the human-normalized score, each variable the median over the 30 runs). Across the seven networks, the distance from isometry correlates with the score at -0.79, -0.75 and -0.50 under Muon, RAdam and Kron, and the rank of the features at $+0.89$, $+0.46$ and $+0.99$. The gradient norm correlates at -0.89, -0.25 and -0.71, the top eigenvalue of the Hessian of the loss at -0.36, $+0.32$ and -0.71, its trace at -0.07, $+0.25$ and -0.71, and the weight norm at $+0.96$, $+0.07$ and $+0.89$: their sign or their strength depends on the optimizer. Figure 18 shows that the simplicity score of Lee et al. (2025a) at initialization does not separate the variants of our layer.

Over training. Figure 13 follows the constant c_k of Proposition 1 over training, from nine checkpoints of every run on all ten games with three seeds. The mean constant starts at 1.0 and grows during training under all three optimizers, to 1.84 under Muon, 1.74 under RAdam and 2.68 under Kron at depth 32. At depth 32, $\max_k c_k/K$ ends at 0.06 under Muon and 0.10 under RAdam and Kron on average, and at 0.23 at most in any run (BattleZone under Kron), so the assumption $c_k \le K/2$ holds at every state and checkpoint at depths 10 and 32; at depth 32 the measured distance from isometry stays below a sixth of what the bound allows. At depth 2 this assumption is violated from the first update on ($\max_k c_k$ reaches about 1.9, against $K/2 = 1$), although the inequality of Proposition 1 still holds at every measured state. Two further analyses use a separate set of runs with nine checkpoints, on four of the ten games with one seed. The covariance of the features that reach the output layer becomes better conditioned with depth for our network and worse for the plain network, and under RAdam our weight matrices grow 34-fold in norm while the condition number

Table 10: **Craftax-Symbolic**, PQN at 100M steps, three seeds. *Left*: episode return over the last 10% of training (at most 226); the published network scores 17.0 at its published learning rate, and 12.4 and 11.3 at three and ten times that rate. *Right*: percentage of episodes in which an achievement is reached at the end of training.

Depth	Return
1	17.6
2	20.3
4	21.8
8	22.3
16	22.9
32	23.2
Baseline	17.0

	Ours, depth		
Achievement	1	32	Baseline
Collect iron	6.4	45.7	18.5
Enter the dungeon	29	45	11
Defeat an orc mage	0.7	9.4	0.1
Make a stone pickaxe	45	75	82
Eat a cow	–	80	95

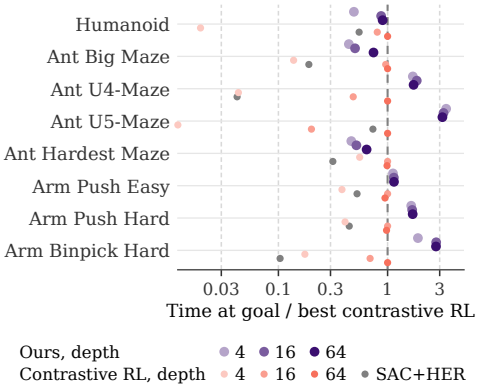

Figure 7: **Goal-conditioned RL** on eight JaxGCRL tasks: final time at goal of SAC with hindsight relabeling and our critic at depths 4, 16 and 64 (purple, light to dark), relative to the best of the contrastive agents of Wang et al. (2025) at depths 4, 16 and 64 on the same task (coral; dashed line at 1). Mean over three seeds; learning curves in Figure 8.

of the hidden layers stays at 1.5 (Figure 15). The effective rank of the activations that reach each weight matrix stays constant along our hidden layers, whereas the plain network loses rank in both the forward and the backward direction (Figure 16). The multi-skip network keeps a high backward rank, but its spectrum carries a gain of $e^{2.5}$ to $e^{2.8}$, which makes up nearly all of its distance from isometry in Figure 12: a rank counts directions, whereas signal propagation depends on the singular values.

Proposition 1 numerically. Figure 17 compares the condition number of a product of K random matrices with that of a stack of K of our layers with the same matrices in their branches: at depth 64 the first reaches 10^{69} for matrices of condition number 100, the second 1.4.

J Ablations and design alternatives

Table 15 gives the interventions against loss of plasticity of Section 3.1 at every depth, at their best learning rates. At depth 10 they recover part of the score only at a tenth of the published learning rate, where the plain network recovers as much.

Table 16 removes the components of our layer one at a time, Table 14 replaces its skip connection by fixed matrices (Section 5.2), and Table 17 collects further controls.

Without the skip connection, our layer becomes a plain stack of normalized layers, since the $1/K$ multiplier is removed with the skip. At depth 32 it fails like the plain network of the same depth: it scores at the level of a random policy under RAdam and Kron, and 0.26 under Muon, against 0.05 for the plain network; with ReLU instead of the shifted ReLU, it scores 0.13 under Muon (0.125 [0.11, 0.15]), half as much. Under Kron its weights also become non-finite late in training, after it has failed to learn (0.02 before that point): the threshold of its last shifted ReLU is followed by a ReLU and receives no gradient, and in the implementation of Kron that we use, the preconditioner of a parameter whose gradient is exactly zero grows until it overflows. Unless stated otherwise, the runs below are PQN on Atari-10 at depth 32 with three seeds; some alternatives were only screened on four games with two seeds, which we state where it applies.

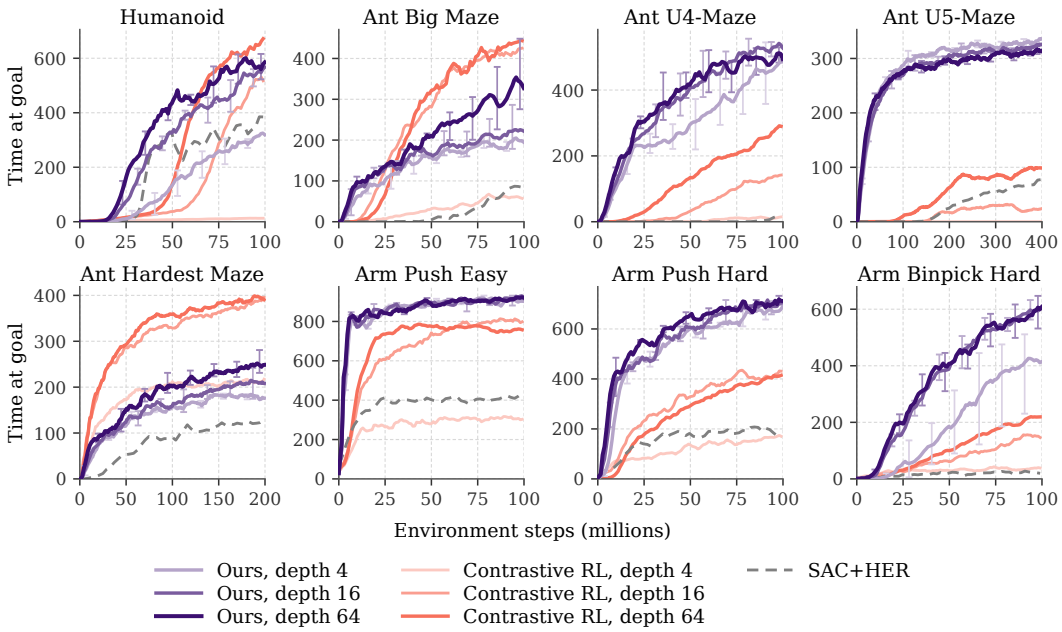

Figure 8: **Goal-conditioned RL**: learning curves behind Figure 7, time at goal (out of 1,000) against environment steps; SAC with hindsight relabeling and our critic at depths 4, 16 and 64 (purple, mean and range over three seeds at eight points), the contrastive agent of Wang et al. (2025) at depths 4, 16 and 64 (coral) and their SAC with hindsight relabeling (grey dashed), from their published results.

Table 11: **Goal-conditioned RL**: time at goal at the end of training (out of 1,000), mean of the last five evaluations. Ours: SAC with hindsight relabeling and our critic, mean over three seeds. Contrastive RL and SAC+HER: the published results of Wang et al. (2025).

	Ours, depth			Contrastive RL, depth			
Task (budget)	4	16	64	4	16	64	SAC+HER
Humanoid (100M)	319	567	585	13	522	649	356
Ant Big Maze (100M)	195	222	328	61	424	441	84
Ant U4-Maze (100M)	486	528	494	12	138	286	12
Ant U5-Maze (400M)	335	317	310	0	20	98	72
Ant Hardest Maze (200M)	182	203	252	217	392	387	123
Arm Push Easy (100M)	902	918	921	308	805	762	423
Arm Push Hard (100M)	690	705	712	171	419	410	187
Arm Binpick Hard (100M)	415	609	607	38	151	219	23

Branch weight $1/\sqrt{K}$. Initialization theory suggests a branch weight of order $1/\sqrt{K}$ (Section 3.2). At depth 32 under Muon (four games, two seeds), a branch weight of $1/\sqrt{2K}$ scores below $1/K$ on all four games, and its trained layers deviate 3.6 times further from the identity.

No branch weight, and a learned one. The multiplier keeps the function of the hidden layers near the skip path, which restricts the function class. Under a non-stationary objective, however, the network must keep adapting as the targets change, which requires the conditioning that the multiplier bounds (Proposition 1). Without the multiplier, the network trains without diverging but scores 0.482 against 0.985, and the distance of its hidden layers from isometry is 1.76 against 0.21. A learnable branch weight in the manner of ReZero (Bachlechner et al., 2021), initialized at $1/K$, reaches 0.98 of the score of the fixed weight at depth 9 (one seed). We keep the fixed weight, which is simpler and does not couple the forward scale of a branch to its learning speed.

Normalization. The branch must be normalized, but the form of the normalization matters little: removing it costs 17% (0.816 against 0.985), whereas LayerNorm in place of the parameter-free RMS normalization is at parity (0.979). Lyle et al. (2024) likewise find that the benefits of layer

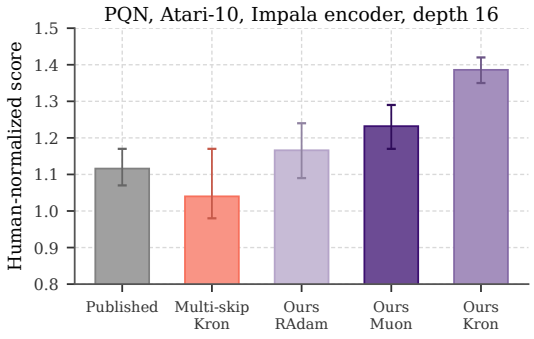

Figure 9: **PQN on Atari-10 behind the Impala encoder** at depth 16: IQM of the human-normalized score with 95% confidence intervals. Published: the published head behind the same encoder.

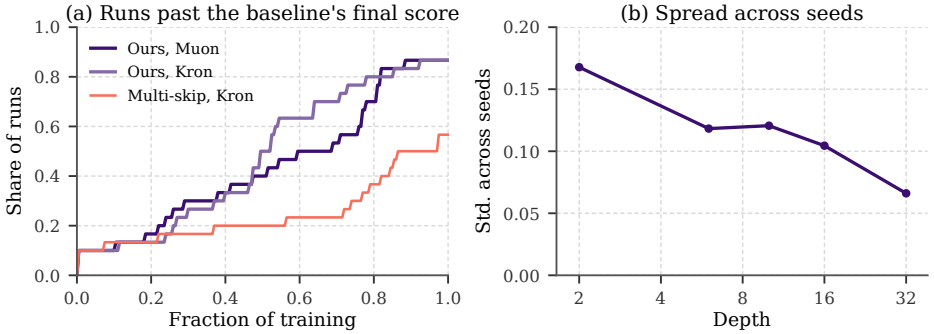

Figure 10: **Learning speed and spread across seeds** with PQN on Atari-10. **(a)** Share of the 30 runs at depth 32 that have reached the final score of the published network, against the fraction of training. **(b)** Our network under Muon: standard deviation of the score across seeds, averaged over the ten games, against depth.

normalization for plasticity come from rescaling the pre-activations, while centering them has no noticeable effect. RMS normalization (Zhang & Sennrich, 2019) keeps only the rescaling, performs comparably to LayerNorm at lower cost, and is the normalization of most recent large language models (Touvron et al., 2023). We use its parameter-free form, since the branch needs no gain (Section 4), and because it is the form that Proposition 1 uses.

Activation. The shifted ReLU follows from the same principle as the rest of the layer: its Jacobian is the identity on every unit above the threshold, so that with a threshold of -1 about five in six normalized inputs pass unchanged, where ReLU passes half (Qi et al., 2020). Inside the full layer, where the skip connection and the $1/K$ multiplier already keep the layer close to an isometry, it gives a modest gain: ReLU and GELU score 0.942 and 0.932 against 0.985, and the exactly norm-preserving Householder activation (Singla et al., 2022) 0.941, at 5% more compute per step; on SAC the Householder activation is lower at depth 32 (1.74 against 2.24). Without the skip connection, where the branch alone carries the signal, the shifted ReLU doubles the score of ReLU (0.259 [0.23, 0.29] against 0.125 [0.11, 0.15]) and is higher on 9 of the 10 games. An initial threshold of -2 instead of -1 scores alike (0.994), and 0 somewhat lower (0.917; Table 17). We therefore keep the shifted ReLU: it adds a modest gain at no cost, and it serves the same aim as the other components, a layer whose Jacobian stays close to an isometry.

Initialization. We initialize the weight matrices orthogonally. With suitable activations, orthogonal weights give dynamical isometry in deep networks where Gaussian weights cannot (Saxe et al., 2014; Pennington et al., 2017), and orthogonal initialization has become the default of widely used deep RL implementations such as the PPO agent of CleanRL (Huang et al., 2022). In our layer it also makes the constants c_k of Proposition 1 close to one at initialization. An initialization sets the Jacobian only at the start of training; the rest of the layer is designed to keep it close to an isometry throughout training.

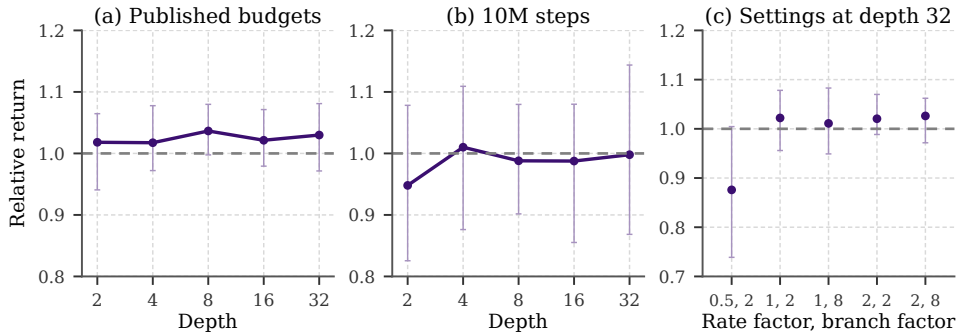

Figure 11: **PPO on the Playground locomotion tasks**, one seed per task: IQM of the return relative to the published agent (dashed), with 95% confidence intervals over tasks. **(a)** Our value network against depth at the published budgets (16 tasks). **(b)** The same at 10M environment steps. **(c)** Five learning-rate and branch learning-rate factors at depth 32 (13 tasks).

Table 12: **Training time per run**: median wall-clock hours of one run (one seed on one game or task, at the budgets of Tables 4 and 5) on one NVIDIA H200 GPU, over the 23 to 48 runs of each network and depth. In PPO the multi-skip network deepens both the policy and the value network, ours only the value network.

PQN, Atari-10						
Depth	0	2	6	10	16	32
Ours, Muon		0.16	0.17	0.19	0.21	0.28
Ours, RAdam		0.14	0.15	0.16	0.17	0.21
Ours, Kron		0.16	0.18	0.20	0.23	0.31
Multi-skip, Kron		0.17	0.19	0.21	0.25	0.34
Plain, RAdam		0.14	0.16	0.17	–	0.25
Published network, RAdam	0.14					
SAC, DeepMind Control						
Depth	1	2	4	8	16	32
Ours		0.23	0.27	0.37	0.56	0.95
SimbaV2		0.26	0.30	0.39	0.58	0.91
Published network	0.19					
PPO, DeepMind Control						
Depth		2	4	8	16	32
Ours		0.11	0.13	0.15	0.23	0.33
Multi-skip		0.14	0.17	0.24	0.36	0.60
Published network			0.11			

An orthogonality penalty in place of Muon. Muon keeps our weight matrices close to orthogonal (median condition number 6.1 at the end of training, against 2,600 under RAdam and 3,069 under Kron). A first-order optimizer with a soft orthogonality penalty $\frac{1}{2}\lambda\|\boldsymbol{W}^\top\boldsymbol{W} - s\boldsymbol{I}\|_F^2$ (Cisse et al., 2017; Chung et al., 2024), applied as a decoupled step, also keeps the weights orthogonal (condition number 2.3 to 3.8), but reaches only 0.69 to 0.76 of the score of the Muon network over two learning rates and two penalty strengths (four games, two seeds). The benefit of Muon therefore does not come from the orthogonality of the weights alone.

Skip connections and updates per step. One skip connection per weight matrix matters: with a skip connection every four layers, at equal parameters, the score falls to 0.788. On SAC, 2 instead of 8 updates per environment step lower the score at depth 32 (1.32 against 2.24), but the interaction between depth and the number of updates, $+0.32$ $[-0.11, +0.70]$, is not distinguishable from zero.

The plain network. Halving or doubling the learning rate does not rescue the plain network on Atari. Spectral normalization (Bjorck et al., 2021), which bounds only the largest singular value of each weight matrix, does not rescue the plain SAC critic at depth 32 and harms it at depth 8.

Table 13: **All measured quantities against the score**: Spearman correlation between each quantity at the end of training and the final score across all runs of all networks and optimizers, within each game or task, averaged (1,565 runs). Table 1 gives 95% intervals for the main quantities.

	Atari, PQN	DMC, PPO	DMC, SAC
Jacobian of the hidden layers			
Share of directions within a factor 2 of isometry	+0.61	+0.32	+0.27
Distance from isometry	−0.53	−0.34	−0.26
Effective rank	+0.48	+0.16	+0.40
Condition number (log)	−0.44	−0.05	−0.18
Fraction of lost directions	−0.49	−0.33	−0.25
Mean squared singular value	+0.43	+0.31	+0.21
Tangent kernel of the hidden layers			
Effective rank	+0.32	+0.39	+0.26
Scale (trace)	−0.01	−0.08	−0.03
Features and other quantities			
Feature rank	+0.53	+0.18	+0.42
Dormant units	−0.14	−0.25	−0.37
Weight norm	+0.30	+0.25	+0.24
Gradient norm	+0.05	+0.01	−0.07
Largest Hessian eigenvalue	+0.14	–	–

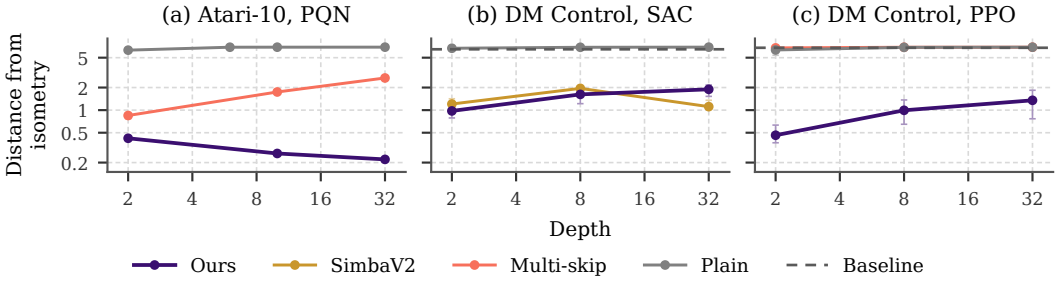

Figure 12: **Distance from isometry at the end of training** against depth: root mean square of the logarithms of the singular values of the Jacobian of the hidden layers, with values below 10^{-3} counted as lost (at most $\ln 1000 \approx 6.9$, when all are lost). Median over runs, bars from the first to the third quartile; dashed: the published network.

K Continual supervised learning

Figure 19 reproduces the supervised experiment of Castanyer et al. (2025) with their code. On the true labels every network fits. With the labels reshuffled every 20 epochs, the plain network at depth 10 does not recover after the first reshuffle and the one at depth 6 recovers only partly and late, and the plain networks at depths 6 and 10 cannot fit one fixed random labelling from the start: their features collapse to rank one within six epochs. Our network fits the random labelling at every depth, although more slowly than the 20 epochs between two reshuffles allow.

The remainder of this section uses a smaller setup that isolates the hidden layers.

The two protocols share the following setup.

- *Random labels.* A fixed subset of 4,096 CIFAR-10 training images. Each of twenty tasks draws new uniformly random labels for these images; we report the train accuracy at the end of each task.
- *Permuted pixels.* The CIFAR-10 training and test sets. Each of twenty tasks applies a new random permutation to the 3,072 input coordinates; we report the test accuracy at the end of each task.
- *Tasks.* Every task lasts 8,000 steps, twice the time a freshly initialized two-layer network needs to reach 95% of its final accuracy.

Table 14: **Frozen skips on Atari-10**, our layer at depth 32: score relative to the identity skip under the same optimizer, median over the ten games of the per-game ratio (games better / within 5% / worse). Three seeds per game.

Skip path of every layer	Muon	RAdam	Kron
Identity	$\times 1.00$	$\times 1.00$	$\times 1.00$
Frozen random orthogonal	$\times 0.98$ (2/5/3)	$\times 1.05$ (5/3/2)	$\times 1.00$ (4/5/1)
Frozen, singular values in $[0.43, 1.73]$	$\times 0.93$ (1/1/8)	$\times 0.86$ (3/1/6)	$\times 1.03$ (5/4/1)
Frozen, singular values in $[0.15, 2.37]$	$\times 0.73$ (2/2/6)	$\times 0.83$ (1/1/8)	$\times 0.98$ (2/5/3)
Frozen Gaussian	$\times 0.82$ (1/2/7)	$\times 0.83$ (0/3/7)	$\times 0.98$ (4/2/4)
No $1/K$ multiplier	$\times 0.54$ (1/1/8)	$\times 0.55$ (0/0/10)	$\times 0.81$ (0/1/9)
No skip	$\times 0.31$ (0/1/9)	$\times -0.01$ (0/0/10)	$\times -0.01$ (0/0/10)

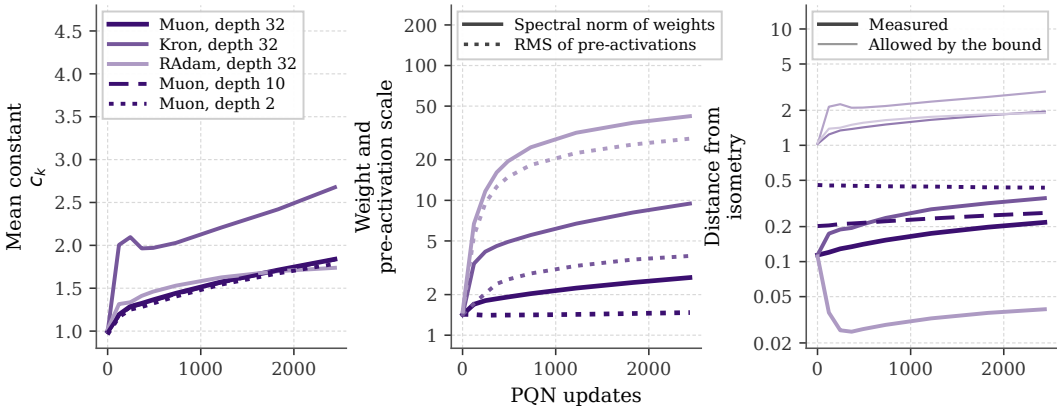

Figure 13: **The constant of Proposition 1 on Atari**, PQN, averaged over layers and over ten games and three seeds. *Left*: mean constant c_k over training. *Middle*: its two factors at depth 32, the spectral norm of the weights and the RMS of the pre-activations. *Right*: distance of the hidden layers from isometry, measured and allowed by the bound.

- *Networks.* A linear input layer to width 256, $K \in \{2, 4, 8, 16, 32\}$ hidden layers, and a linear output layer, so that every network has $K + 2$ weight matrices (Table 18).
- *Training.* Mean squared error on one-hot targets, Adam without weight decay, batch size 256, a constant learning rate, three seeds.
- *Learning rate.* For each network, the rate with the highest final train accuracy in a stationary run of 10,000 steps with the true labels at depth 32, chosen from $\{3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}\}$ and extended in the same steps until the best rate lies inside the range. The same rate is used at every depth and in both protocols.
- *Measurements.* At the end of every task, on a fixed batch of 64 images: the singular values of the Jacobian of the hidden layers (from the input of the first hidden layer to the output of the last) and the ratio of output to input distances over pairs of images.

Table 18 gives the scores of all eighteen networks. The ReLU network with He initialization fits the first six random-label tasks at depth 32 and then falls to chance, and over the last five tasks it is at chance at every depth. Trained for the same 160,000 steps on the stationary task with the true labels, it reaches a final train accuracy of 0.99 up to depth 16 and 0.83 at depth 32, and the networks with orthogonal initialization, LayerNorm or tanh reach 0.99 at every depth. The length of training alone therefore does not explain the failure. With orthogonal initialization or LayerNorm it keeps fitting at depth 32, although orthogonal initialization fails at depths 2 and 4. On permuted pixels, all plain ReLU networks lose test accuracy with depth. The Jacobians of their hidden layers are numerically singular on most images from the first task on, so their spectrum carries no further information.

The exactly isometric networks fail differently. The spread of the spectrum of the MaxMin network stays at about 10^{-4} by construction, yet at depth 32 it does not fit even the first random-label task, and its permuted-pixel accuracy falls to 0.21. Distances between inputs shrink instead: on random labels, the fifth percentile of the ratio of output to input distances falls from 0.12 after the first task

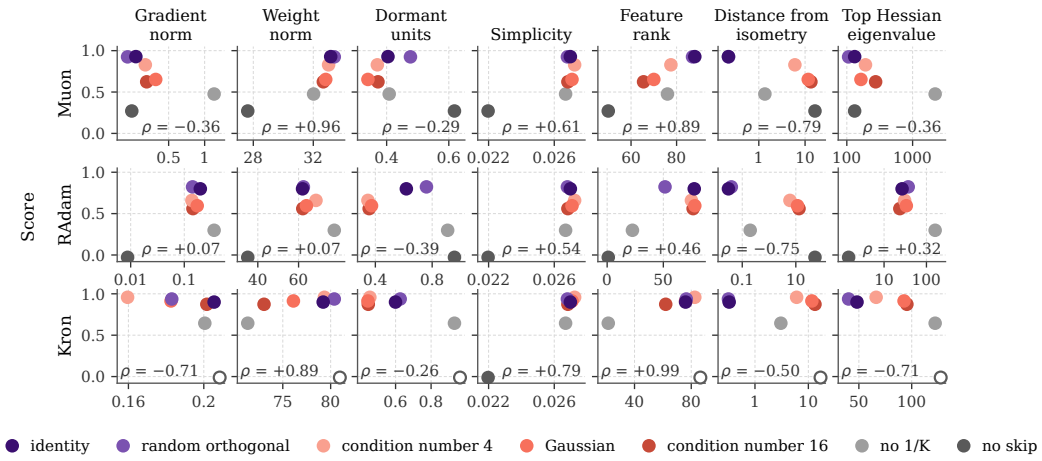

Figure 14: **Competing explanations under three optimizers.** Score against candidate variables for the seven depth-32 variants of our layer on Atari-10 (identity skip, frozen random orthogonal skip, frozen skips of condition number 4 and 16, frozen Gaussian skip, no $1/K$ multiplier, no skip), one row per optimizer. ρ is the Spearman rank correlation between the variable and the score over the variants with a defined value. Without a skip, the network does not learn under Kron and its weights become non-finite late in training (Appendix J), so the measurements of the trained network are undefined; it is drawn as an open circle at the right edge of those panels and ranked last in the distance panel.

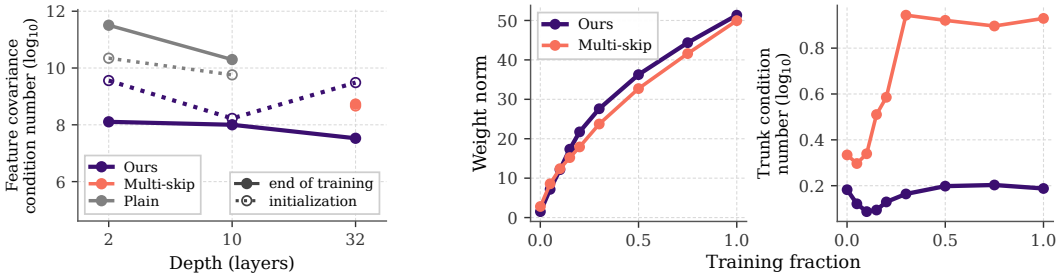

Figure 15: **Feature covariance and weight norms**, four Atari games, one seed. *Left*: condition number of the covariance of the features read by the output layer at the end of training, against depth. *Right*: spectral norm of the weights and condition number of the hidden layers over training under RAdam, both networks at depth 32.

to 0.05 after the last. Appendix C.3 explains how an exactly isometric Jacobian allows this. With a soft orthogonality penalty instead of exact orthogonality, the network reaches 0.71 on random labels at depth 32. The Householder network fits every random-label task, but its permuted-pixel accuracy falls with depth.

Our network fits every random-label task at every depth, and its permuted-pixel accuracy does not fall with depth (0.496 at depth 2, 0.505 at depth 32). All variants of our layer also fit every random-label task, so only permuted pixels separate them. With $1/\sqrt{K}$ instead of $1/K$, or without the multiplier, accuracy falls with depth (to 0.489 and 0.467 at depth 32), and without normalization it is lower at every depth. After twenty tasks, the spread between the 5th and 95th percentiles of the log singular values of the Jacobian of the hidden layers at depth 32 is 3.2 for our layer and 11.6 to 14.6 for the three variants.

With the identity skip replaced by a frozen matrix, all variants score alike at depth 2 (0.50). At depth 32, a frozen random orthogonal skip scores 0.515, whereas frozen skips with singular values spread over $[0.43, 1.73]$ and $[0.15, 2.37]$ and a frozen Gaussian matrix score 0.457, 0.441 and 0.445.

To test priming (Nikishin et al., 2022), we first train each network at depth 32 for 20,000 updates on 1,024 images of five classes, with all-zero targets or with their true labels, then on the full task, and compare it with the same network trained from scratch (Figure 20). Over the first fifth of the

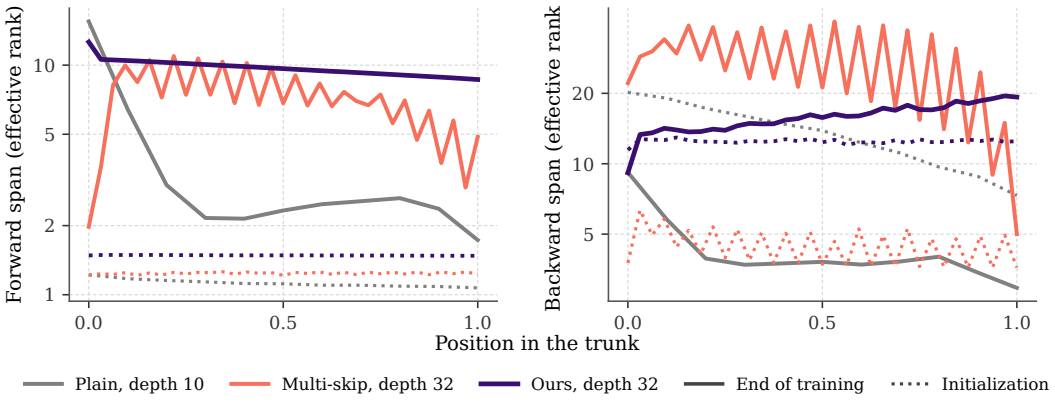

Figure 16: **Forward and backward spans**, four Atari games, one seed. Effective rank of the activations (left) and of the sensitivities (right) that reach each weight matrix of the hidden layers, at the end of training (solid) and at initialization (dotted).

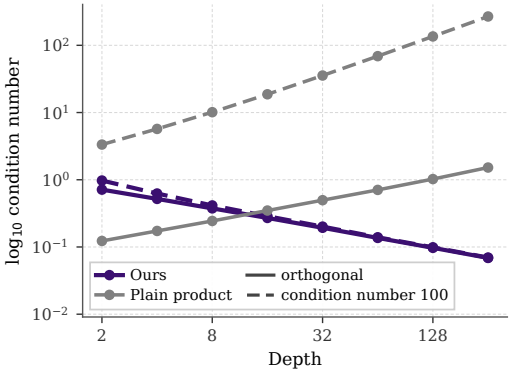

Figure 17: **Proposition 1 numerically.** Condition number of a plain product of K random matrices and of a stack of K of our layers with the same matrices in their branches, for orthogonal matrices and for matrices of condition number 100; median over 20 draws.

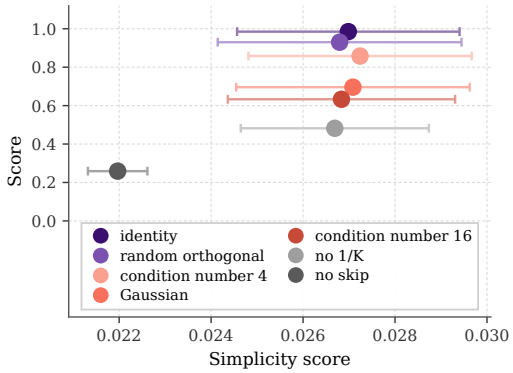

Figure 18: **Simplicity bias.** Simplicity score of Lee et al. (2025a) at initialization for the seven variants of our layer. The six with a skip connection lie within 2% of each other, while their Atari scores range from 0.48 to 0.98.

second phase, the He-initialized ReLU network trails its unprimed version by 0.03 and 0.09 in test accuracy after the two primings, and still by 0.01 and 0.04 at the end. Our network is not slowed down: it is ahead of its unprimed version early on and within 0.01 at the end. The ReLU network with orthogonal initialization is not slowed down either, and the MaxMin network, whose unprimed version learns slowly at this depth, benefits from priming. The tanh network with orthogonal initialization, which is close to an isometry only while its units stay in the linear regime, is slowed down by priming on the true labels about as much as the He-initialized ReLU network (0.12 early, 0.03 at the end), consistent with isometry at initialization being insufficient.

L PER-TASK RESULTS

Tables 19 to 22 give the scores behind the aggregates.

Table 15: **Interventions for plasticity at depth**, PQN on Qbert / Phoenix, human-normalized score, mean of three seeds, at the best learning rate of the sweep (in brackets, relative to the published 2.5×10^{-4}; the sweep covers $\times 0.1$ to $\times 3$ at depths 10 and 32). The deepened PQN network (LayerNorm and ReLU) with normalize-and-project (Lyle et al., 2024) and with spectral regularization (Lewandowski et al., 2025). Ours at depth 32 under RAdam scores 0.94 / 2.14.

Network	Depth 2	Depth 10	Depth 32
Plain, RAdam	0.55 / 1.16 ($\times 1$)	0.13 / 0.53 ($\times 0.1$)	0.01 / -0.09 ($\times 0.1$)
NaP	0.22 / 0.63 ($\times 1$)	0.07 / 0.54 ($\times 0.1$)	0.01 / -0.09 ($\times 0.1$)
Spectral regularization	0.37 / 0.62 ($\times 1$)	0.15 / 0.58 ($\times 0.1$)	0.02 / -0.09 ($\times 0.1$)

Table 16: **Ablation** on Atari-10 with PQN at depth 32 under Muon: one change to the full layer at a time, IQM [95% CI] over 30 runs and per-game wins-ties-losses (5% band).

Change	IQM [95% CI]	W-T-L
Full layer	0.985 [0.95, 1.01]	
RMS norm $\rightarrow$ LayerNorm	0.979 [0.95, 1.02]	1-5-4
Shifted ReLU $\rightarrow$ ReLU	0.942 [0.91, 0.98]	1-6-3
Shifted ReLU $\rightarrow$ GELU	0.932 [0.90, 0.96]	
Muon $\rightarrow$ RAdam	0.802 [0.73, 0.87]	2-3-5
No normalization	0.816 [0.75, 0.90]	1-2-7
$1/K \rightarrow 1/\sqrt{K}$ multiplier	0.832 [0.78, 0.88]	0-2-8
No $1/K$ multiplier	0.482 [0.46, 0.50]	1-1-8
No skip	0.259 [0.23, 0.29]	0-1-9
Published network, RAdam	0.790 [0.73, 0.87]	2-1-7

Table 17: **Controls.** Each row changes one thing in the configuration named in the group heading, without re-tuning, at the same depth: IQM [95% CI]. The plain SAC critic uses two seeds.

Change	IQM [95% CI]	Reference
PQN on Atari-10, ours at depth 32 under Muon		
Weight norm held at its initial value	0.981 [0.945, 1.016]	0.985 [0.953, 1.015]
One skip connection every four layers	0.788 [0.719, 0.875]	0.985 [0.953, 1.015]
Shifted-ReLU threshold 0 at initialization	0.917 [0.864, 0.971]	0.985 [0.953, 1.015]
Shifted-ReLU threshold -2 at initialization	0.994 [0.940, 1.047]	0.985 [0.953, 1.015]
Householder activation	0.941 [0.88, 1.00]	0.985 [0.953, 1.015]
PQN on Atari-10, plain network under RAdam		
Depth 6, half the learning rate	0.200 [0.190, 0.212]	0.100 [0.066, 0.136]
Depth 6, twice the learning rate	0.014	0.100 [0.066, 0.136]
Depth 10, half the learning rate	0.013 [0.007, 0.018]	0.021 [0.012, 0.032]
Depth 10, twice the learning rate	-0.007	0.021 [0.012, 0.032]
SAC on DeepMind Control, ours at depth 32		
Frozen random orthogonal skip	2.00 [1.66, 2.24]	2.24 [2.17, 2.38]
Frozen skip with singular values in $[0.15, 2.37]$	1.75 [1.45, 2.27]	2.24 [2.17, 2.38]
2 instead of 8 updates per environment step	1.32 [1.14, 1.67]	2.24 [2.17, 2.38]
Householder activation	1.74 [1.43, 2.04]	2.24 [2.17, 2.38]
SAC on DeepMind Control, plain critic with the training improvements		
Spectral normalization, depth 8	0.11 [0.10, 0.12]	1.81 [1.72, 1.92]
Spectral normalization, depth 32	0.06 [0.05, 0.06]	0.05 [0.05, 0.06]

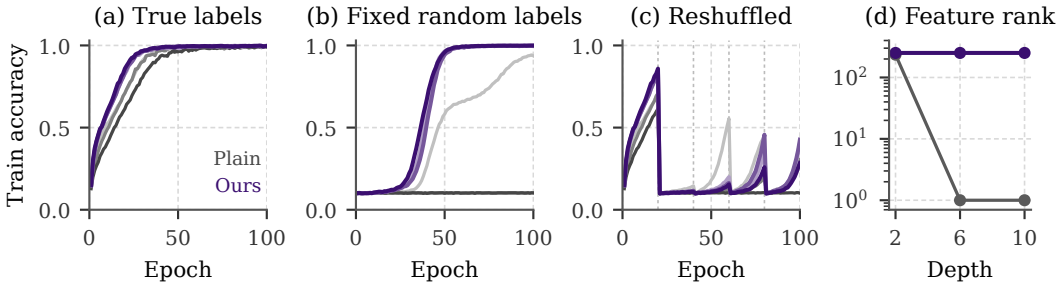

Figure 19: **CIFAR-10 with the code and protocol of Castanyer et al. (2025)**: a convolutional encoder followed by plain fully connected layers (grey, darker is deeper) or by our layers (purple), at depths 2, 6 and 10, three seeds. **(a)–(c)** Train accuracy per epoch on the true labels, on one fixed random labelling, and with the labels reshuffled every 20 epochs (dotted lines), the protocol of Castanyer et al. (2025). **(d)** Rank of the features that reach the output layer at the end of (b).

Table 18: **Continual supervised learning**, three seeds. Random labels: train accuracy, mean over the last five of twenty tasks (chance is 0.10). Permuted pixels: test accuracy, mean over the last five tasks. Spread: difference between the 95th and 5th percentiles of the logarithms of the singular values of the Jacobian of the hidden layers at depth 32 after the last permuted-pixel task; *singular*: the Jacobian of the hidden layers is numerically singular, > 16: beyond the numerical resolution of the decomposition.

		Random labels			Permuted pixels			
Network	Learning rate	2	8	32	2	8	32	Spread
Plain								
ReLU, He initialization	$3 \cdot 10^{-4}$	0.11	0.11	0.11	0.468	0.450	0.450	singular
ReLU, orthogonal initialization	10^{-4}	0.11	1.00	0.92	0.449	0.443	0.442	singular
Shifted ReLU, orthogonal initialization	$3 \cdot 10^{-4}$	0.10	0.17	0.10	0.466	0.447	0.363	singular
ReLU with LayerNorm	10^{-4}	1.00	1.00	0.98	0.494	0.457	0.446	13.4
tanh, orthogonal initialization	10^{-4}	1.00	1.00	0.11	0.472	0.464	0.460	7.4
Leaky ReLU, deep kernel shaping	10^{-4}	0.98	1.00	1.00	0.502	0.516	0.534	11.6
Exactly isometric								
MaxMin	10^{-3}	1.00	1.00	0.11	0.489	0.508	0.206	0.0
MaxMin, soft orthogonality penalty	$3 \cdot 10^{-4}$	1.00	1.00	0.71	0.478	0.499	0.468	0.1
Householder	10^{-5}	1.00	1.00	1.00	0.484	0.404	0.429	0.0
Residual								
Pre-LayerNorm ReLU residual	10^{-3}	1.00	1.00	0.96	0.472	0.475	0.467	> 16
Ours	$3 \cdot 10^{-3}$	1.00	1.00	1.00	0.496	0.491	0.505	3.2
$1/\sqrt{K}$ instead of $1/K$	$3 \cdot 10^{-4}$	1.00	1.00	1.00	0.508	0.503	0.489	11.6
no $1/K$ multiplier	10^{-3}	1.00	1.00	1.00	0.500	0.492	0.467	14.6
no normalization	$3 \cdot 10^{-3}$	1.00	1.00	1.00	0.465	0.468	0.469	13.8
Ours with a frozen skip								
random orthogonal	$3 \cdot 10^{-3}$	1.00	1.00	1.00	0.502	0.516	0.515	1.7
condition number 4	$3 \cdot 10^{-3}$	1.00	1.00	1.00	0.502	0.472	0.457	13.6
condition number 16	$3 \cdot 10^{-3}$	1.00	1.00	1.00	0.501	0.468	0.441	> 16
Gaussian	$3 \cdot 10^{-3}$	1.00	1.00	1.00	0.499	0.467	0.445	> 16

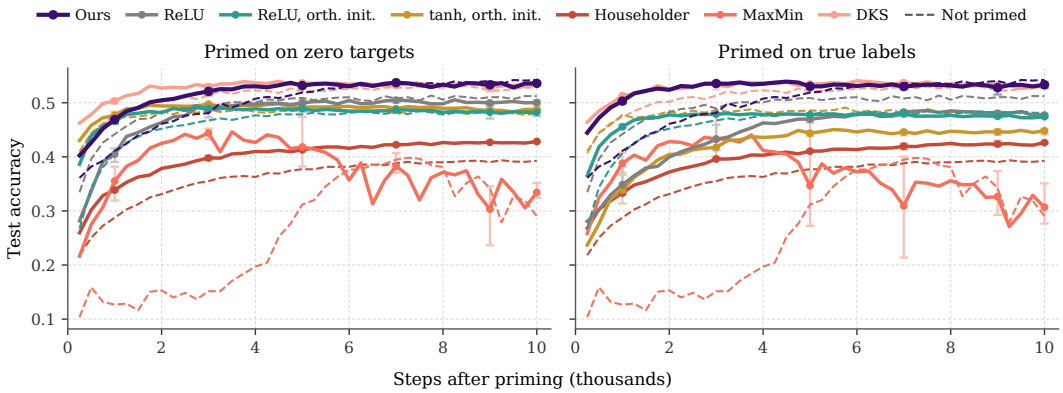

Figure 20: **Priming.** Test accuracy on the full task after 20,000 updates on 1,024 images of five classes with all-zero targets (left) or with their true labels (right), at depth 32, against the same network trained from scratch (dashed). ReLU: He initialization; ReLU, orth. init.: the same network with orthogonal initialization; Householder and MaxMin: exactly orthogonal weight matrices; DKS: deep kernel shaping with leaky ReLU.

Table 19: **Atari-10 per game**, PQN at 40M frames: human-normalized score, mean over three seeds, for the Baseline, the plain network at depth 10, the multi-skip network at depth 32 and ours at depth 32, grouped by optimizer. The last rows give the IQM over the 30 runs and its 95% confidence interval.

	RAdam			Muon		Kron	
Game	Baseline	Plain	Ours	Baseline	Ours	Multi-skip	Ours
Amidar	0.2	0.0	0.2	0.2	0.2	0.2	0.2
Battle Zone	0.8	0.0	0.7	0.7	0.7	0.6	0.8
Bowling	0.1	0.0	0.1	0.1	0.1	0.1	0.1
Double Dunk	7.8	6.2	7.5	7.3	7.7	7.5	7.6
Frostbite	0.5	0.0	0.6	0.7	0.7	0.7	0.8
Kung-Fu Master	1.1	0.0	1.2	1.2	1.5	0.9	1.1
Name This Game	1.3	−0.2	0.8	1.7	1.7	0.6	0.9
Phoenix	1.9	−0.1	2.4	2.2	2.3	0.7	2.0
Q*bert	0.6	0.0	0.9	0.6	0.8	0.9	1.1
River Raid	0.7	0.3	0.6	0.5	0.6	0.5	0.8
IQM	0.790	0.021	0.802	0.902	0.985	0.636	0.915
95% CI	0.73–0.86	0.01–0.03	0.73–0.87	0.87–0.94	0.95–1.01	0.62–0.65	0.88–0.94

Table 20: **SAC on DeepMind Control, per task**: final evaluation return divided by the published return (first column), mean over three seeds; a diverged run scores zero. The last row gives the IQM over all runs.

			Ours, depth			SimbaV2, depth		
Task	Published	Baseline	2	8	32	2	8	32
AcrobotSwingup	41.2	4.76	5.77	8.95	9.95	3.46	5.22	7.41
FingerTurnHard	731.5	0.84	1.09	1.28	1.32	1.16	1.30	1.29
FishSwim	460.2	0.79	0.87	0.95	1.14	0.56	1.08	1.19
HopperHop	121.9	2.64	1.53	3.24	3.72	2.26	1.95	1.70
HopperStand	155.7	4.24	4.89	4.88	4.26	5.04	5.09	3.80
HumanoidRun	228.3	1.38	2.20	2.74	2.91	0.32	1.81	1.34
SwimmerSwimmer6	416.0	1.17	0.87	1.57	1.33	0.65	0.97	1.32
WalkerRun	710.9	1.14	1.18	1.23	1.24	1.11	1.20	1.21
IQM		1.51	1.55	2.02	2.24	1.33	1.57	1.59

Table 21: **SAC on locomotion, per task**: mean of the last three evaluation returns divided by that of the published SAC agent in our runs (first column), mean over three seeds. The last row gives the IQM over all runs.

			Ours, depth			SimbaV2, depth		
Task	Reference	Baseline	2	8	32	2	8	32
Go1Footstand	9.64	1.32	1.52	1.61	1.66	1.35	1.51	1.62
Go1Getup	6.20	1.56	1.68	2.59	2.59	2.55	2.60	2.62
Go1Handstand	8.38	1.40	1.45	1.57	1.53	1.39	1.35	1.52
Go1JoystickFlatTerrain	4.31	1.60	2.63	2.91	3.10	1.15	2.67	2.72
Go1JoystickRoughTerrain	4.34	1.47	2.60	2.83	3.10	1.21	2.43	2.96
H1InplaceGaitTracking	2.38	1.05	1.48	2.99	2.76	1.44	1.20	1.22
H1JoystickGaitTracking	3.22	3.00	2.41	4.87	5.25	2.06	3.62	5.03
Op3Joystick	8.10	1.05	1.32	1.36	1.52	0.94	1.46	1.50
SpotFlatTerrainJoystick	24.45	1.26	1.36	1.63	1.72	1.31	1.69	1.73
SpotGetup	4.51	2.48	3.65	3.33	3.49	2.78	3.74	3.41
SpotJoystickGaitTracking	26.24	1.24	1.29	1.30	1.33	1.25	1.34	1.34
IQM		1.37	1.77	2.26	2.34	1.37	1.90	2.03

Table 22: **PPO on DeepMind Control, per task**: final evaluation return divided by the published return (first column), mean over three seeds ([2]: two seeds); a diverged run scores zero. Ratios far above one occur where the published return is small. The last row gives the IQM over all runs.

			Ours, depth			Multi-skip, depth	
Task	Published	Baseline	4	16	32	2	32
AcrobotSwingup	222.1	1.08	1.41	1.62	1.54	1.77	2.03
AcrobotSwingupSparse	12.1	3.57	3.40	7.86	11.2	3.40	0.00
BallInCup	565.3	1.53	1.71	1.14	1.14	0.00	1.71
CartpoleSwingup	693.2	1.15	1.10	1.12	1.06	1.25	0.97
CartpoleSwingupSparse	157.9	0.84	3.07	4.75	3.21	0.00	0.00
CheetahRun	825.0	1.10	1.11	1.11	1.12	1.03	0.88
FingerSpin	569.9	0.68	1.30	0.98	1.00	1.09	1.32
FishSwim	635.8	1.02	1.04	1.05	1.04	1.08	0.96
HopperHop	2.1	1.09	44.4	57.3	26.3	44.7	5.45
HopperStand	205.1	0.34	2.63	2.74	2.64	1.98	1.05
HumanoidRun	146.1	1.01	0.51	0.36	0.32	0.07	0.06[2]
HumanoidStand	795.5	1.10	0.85	0.46	0.27	0.06	0.05
HumanoidWalk	468.8	1.63	0.46	0.18	0.16	0.06	0.04
PendulumSwingup	62.9	1.10	13.1	13.2	13.2	5.14	0.26
SwimmerSwimmer6	573.7	0.97	1.01	1.04	1.09	1.06	1.05
WalkerRun	601.2	1.07	1.19	1.17	1.23	1.30	0.76
IQM		1.08	1.28	1.36	1.28	0.82	0.66

Table 23: **SAC and PPO on the remaining DeepMind Control tasks**: IQM of the final return divided by the published return of the same algorithm, with 95% confidence intervals in brackets and three seeds per task, over the tasks not used in the main text (15 for SAC, 7 for PPO) and over all 23 tasks; the two Reacher tasks are not included. SimbaV2 uses its published width of 512, and the Baseline keeps the depth of the published network.

	Depth				
Network	2	4	8	16	32
SAC, the 15 tasks outside the main text					
Baseline			1.01 [1.00, 1.02]		
Ours	1.01 [1.00, 1.02]	1.02 [1.01, 1.04]	1.02 [1.01, 1.04]	1.02 [1.01, 1.04]	1.03 [1.01, 1.04]
SimbaV2	1.02 [1.01, 1.03]	–	1.02 [1.01, 1.03]	1.03 [1.01, 1.04]	1.02 [1.01, 1.03]
SAC, all 23 tasks					
Baseline			1.06 [1.03, 1.10]		
Ours	1.04 [1.02, 1.07]	1.11 [1.07, 1.17]	1.12 [1.08, 1.17]	1.14 [1.08, 1.20]	1.16 [1.11, 1.22]
SimbaV2	1.09 [1.06, 1.13]	–	1.12 [1.08, 1.17]	1.13 [1.09, 1.18]	1.10 [1.08, 1.14]
PPO, the 7 tasks outside the main text					
Baseline			1.00 [1.00, 1.00]		
Ours	1.00 [1.00, 1.00]	1.00 [1.00, 1.00]	1.00 [1.00, 1.00]	1.00 [1.00, 1.00]	1.00 [1.00, 1.00]
PPO, all 23 tasks					
Baseline			1.04 [1.02, 1.06]		
Ours	1.04 [1.02, 1.06]	1.09 [1.06, 1.15]	1.06 [1.04, 1.10]	1.08 [1.05, 1.12]	1.08 [1.05, 1.12]