Stochastic Collapse Environments: A Benchmark for Agentic AI Research

Advait Parulekar
advaitp@utexas.edu
The University of Texas at Austin
Austin, Texas, USA

Alexandros G. Dimakis
alex@bespokelabs.ai
UC Berkeley and Bespoke Labs
Berkeley, California, USA

Abstract

Simplicity bias or *stochastic collapse* is a phenomenon in neural network optimization: Shallow networks are susceptible to a failure mode wherein training dynamics cause neurons to cluster, leaving large regions of the input domain uncovered. We show how to use this phenomenon to create an ensemble of tasks for evaluating the research capabilities of AI agents. Our tasks require an agent to discover non-standard training procedures, or get trapped in local minima with high probability. We specifically know that adding a repulsive loss term inspired by mean-field interacting particle systems is sufficient to solve the problem but stronger AI agents may discover other techniques to successfully fit randomly generated smooth functions with a width-constrained one-hidden-layer ReLU networks. We describe the problem formulation, the underlying failure mechanism, an oracle solution, and a suite of difficulty modulations. Empirical evaluation reveals that current AI agents reliably default to hyperparameter search rather than reconsidering fundamental loss function design, making this benchmark a productive discriminator of reasoning capability.

CCS Concepts

• **Computing methodologies** → **Neural networks**; *Reinforcement learning*; Artificial intelligence.

Keywords

simplicity bias, stochastic collapse, shallow neural networks, ReLU networks, agentic AI, benchmark, mean-field theory, repulsive loss

ACM Reference Format:
Advait Parulekar and Alexandros G. Dimakis. 2026. Stochastic Collapse Environments: A Benchmark for Agentic AI Research. In *Proceedings of Center for AI Safety Annual Workshop (CAIS '26).* ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Terminal-Bench and Harbor Tasks [7] are becoming the de-facto standard for coding agent environments used for training and evaluation. In this paper we design an ensemble of RL environments to test the ability of AI agents to solve research problems in Machine Learning. We begin with a remarkable observation about neural network training that can be exhibited in a simple setting.

1.1 Simplicity bias:

The training of neural networks via gradient based methods is effective in practice, yet not completely understood. In fact, the conventional wisdom before the deep learning revolution was that 'neural networks trained by gradient descent would fail due to local minima, extremely large weights, and slow convergence.'[8]

Modern Machine Learning practice adopts a hands-off approach to the optimization of neural network based architectures, leaving details about the neurons and their activation patterns relatively unexplored in comparison to statistical evaluations of the function induced by the network. A deceptive failure mode arises in shallow ReLU networks: gradient-based optimization can cause the hidden neurons to collapse toward identical or near-identical configurations. This results in the resulting function being less expressive than what is suggested by the size of the network, producing high estimation error that is extremely resilient to training. The phenomenon has been studied under the names *simplicity bias* [3, 9] Stochastic Collapse [1] and *condensation* [10].

This failure mode is not obvious and practically significant: the training loss decreases monotonically throughout and appears to converge cleanly (fig. 1), so standard monitoring does not reveal any pathology. Crucially, a resolution requires a novel optimization solution.

1.1.1 *Activation-Point Dynamics.*

A one-hidden-layer ReLU network partitions the real line into piecewise linear regions separated by *activation points* at $p_k = -b/w_k$ for each neuron k. The network's ability to model curvature in the target function in a region of input space depends entirely on the existence of activation points in that region - any interval containing no activation point is fit by a linear function.

During training, activation points move in response to gradient signals. The gradient of the loss with respect to b_k and w_k is computed from the residual on the active set $\{x : w_k x + b > 0\}$ of neuron k. If two neurons k and ℓ have similar parameters $w_k \approx w_\ell$, their gradients are similar. As a result, both activation points respond similarly to the same gradient updates and tend to move together rather than apart. This clustering is stable: once two neurons are close, their gradients are correlated, their updates are aligned, and they remain close - in fact, standard optimization noise (stochastic minibatches, learning rate schedules) do not introduce sufficient turbulence to break this degeneracy. Adaptive optimizers such as Adam [4] also fail.

CAIS '26, Austin, TX

ACM ISBN 978-1-4503-XXXX-X/2026/05
https://doi.org/XXXXXXX.XXXXXXX

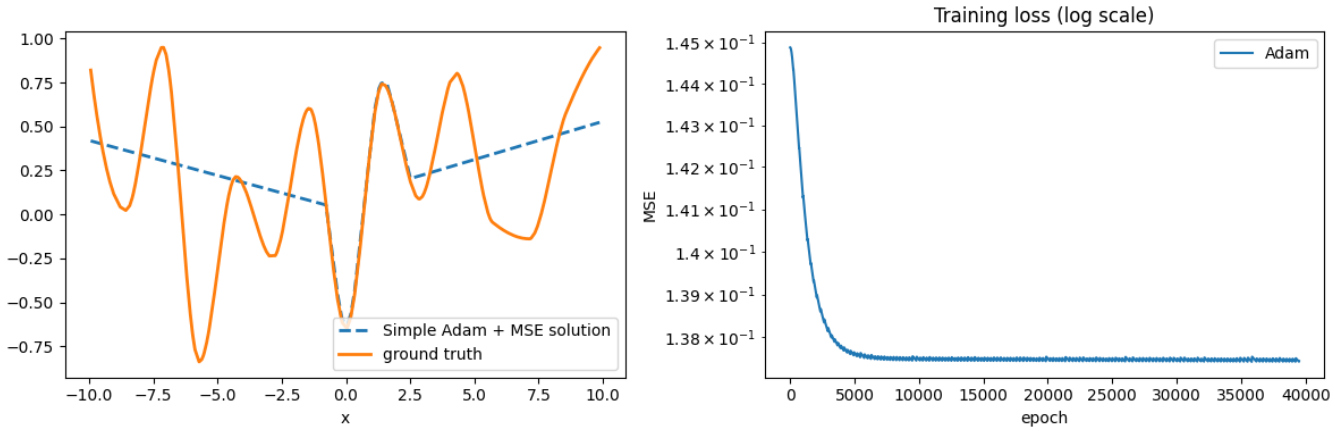

Figure 1: Simplicity bias in practice. ***Left:*** **a width-constrained shallow ReLU network trained with Adam + MSE (dashed blue) converges to a function with very few distinct linear regions, failing to approximate the complex ground truth (orange).** ***Right:*** **the training loss decreases smoothly and converges, giving no indication that the solution is poor.**

2 Task Formulation

We construct environments in which an agent must discover the failure mode of simplicity bias and propose a training procedure that successfully fits random functions drawn from a specified generator using a specified one-hidden-layer ReLU network architecture. The task ensemble is designed to be open-ended, requiring insight into the geometry of the optimization landscape rather than routine hyperparameter tuning.

Regression for Spline Interpolation

We create RL environments that provide the agent with: (1) a description of the random function generator $\mathcal{F}$ (see section 2.2); (2) a fixed network architecture consisting of a single hidden layer of width K and ReLU activation; and (3) an MSE success threshold ϵ. The agent proposes a training procedure $\mathcal{A}$—comprising an optimizer, a loss function, and any associated hyperparameters—which is then used to train 10 independently initialized networks for $T = 7{,}500$ iterations each, with target functions drawn fresh from $\mathcal{F}$. The agent succeeds if at least 5 of the 10 trained networks achieve MSE below ϵ on a held-out evaluation set. Formally, success requires

$$\Pr_{f\sim\mathcal{F},\,\theta_0\sim\text{init}}\left[\text{MSE}(f_{\mathcal{A}(\theta_0,f)}) < \epsilon\right] \geq 0.5.$$

2.1 Network Architecture

The model is a one-hidden-layer ReLU network with K hidden units:

$$f_\theta(x) = \sum_{k=1}^{K} a_k \,\text{ReLU}(w_k x + b),$$

with parameters $\theta = \{(a_k, w_k)\}_{k=1}^{K}, b$. The architecture is fixed; the agent cannot increase K. We refer to the point $\frac{-b}{w_k}$ as the *activation point* of the kth neuron - this is where the ReLU activation transitions from muting the preactivation to letting it through.

2.2 Random Function Generator

Target functions are smooth scalar-to-scalar interpolants. We first sample x_{lim} uniformly from $\{8, 12\}$, and draw $N = 16$ scalars $\{x_i\}_{i=1}^{N}$ uniformly from $[-x_{\text{lim}}, x_{\text{lim}}]$ and sort them. We then draw corresponding output values $\{y_i\}_{i=1}^{N}$ uniformly from $[-1, 1]$. Finally, we interpolate using `scipy.interpolate.PchipInterpolator`, producing smooth curvies with no overshooting. A simple upper bound on the MSE of the best model is just the piecewise linear interpolation of the knots, which has MSE on the order of 10^{-3} for this setup. We refer to the loss of a piecewise-linear interpolation of the knots as MSE_{knot}.

We set the number of knots in the spline interpolation, N, to be one fewer than the number of neurons (that is, the width) K. This ensures that the network has sufficient representation capacity to fit the interolation to "first order" (i.e., with one neuron per knot).

3 The Oracle Solution

3.1 Repulsive Regularization

The oracle solution augments the MSE loss with a *repulsive potential* between pairs of neurons, directly counteracting the collapsing force of gradient descent. Formally, the oracle loss is:

$$\mathcal{L}(\theta) = \frac{1}{M}\sum_{j=1}^{M}\left(f_\theta(x_j) - y_j\right)^2 + \lambda \sum_{k\neq\ell} \frac{1}{\|\hat{\phi}_k - \hat{\phi}_\ell\|^2 + \delta},$$

where $\hat{\phi}_k = (w_k, b)/\|(w_k, b)\|$ is the normalized direction of neuron k's kink-point parameters, $\delta > 0$ is a small numerical stabilizer, and $\lambda > 0$ is a regularization coefficient. Normalization before computing the potential ensures the penalty is scale-invariant and focuses on the angular separation between neurons (i.e., the relative position of activations points on the real line) rather than their absolute magnitude.

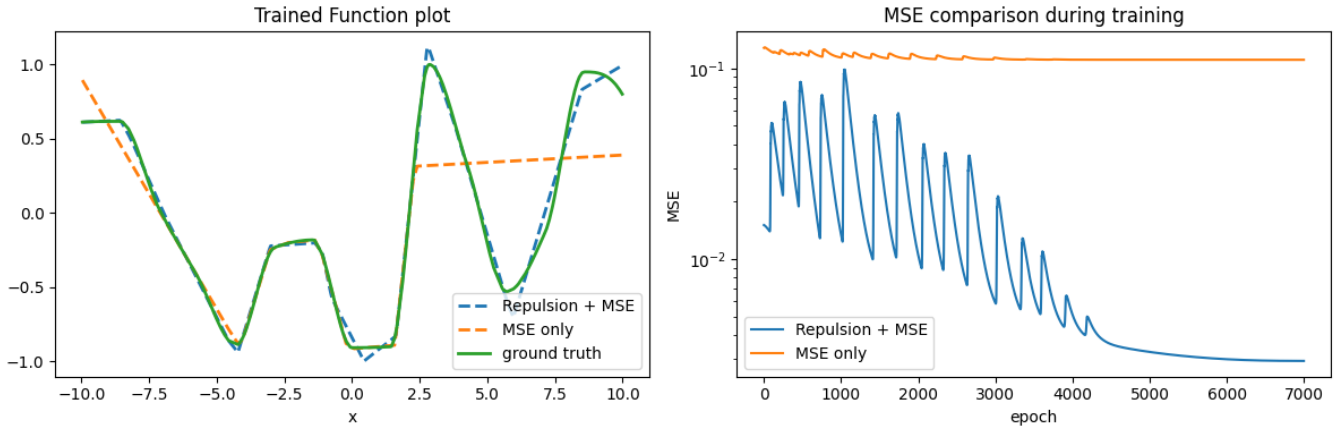

Figure 2: Simplicity bias in practice. ***Left:*** **a width-constrained shallow ReLU network trained with Adam + MSE (dashed orange) fails to approximate the complex ground truth (green). The oracle solution, which uses a potential in the loss corresponding to a repulsion of the neurons, results in training to achieve a better fit (dashed blue)** ***Right:*** **the training loss decreases smoothly and converges, giving no indication that the solution is poor.**

3.2 Training Configuration and Empirical Performance

The oracle training procedure uses the Adam optimizer [4] with a cosine-annealed learning rate schedule [5] over $T = 7{,}500$ iterations. Empirically, the oracle achieves MSE below 10^{-2} on the majority of random function instances with the prescribed network size. A comparison between our oracle and a simple solution comprising of the Adam optimizer on MSE loss is presented in fig. 2.

The oracle fails reliably when ϵ is tightened toward 10^{-4} due to irreducible noise from the random function sampling and the stochasticity of the repulsive landscape near convergence. This provides a natural ceiling for the harder task variants described in Section 4.

4 Benchmark Design and Difficulty Modulation

4.1 Behavior on the Standard Task Ensemble

AI agents consistently fail to innovate beyond the simple Adam + MSE solution. Every agent correctly implements a verifier, which allows the agent to simluate its performance on the task (sufficient information is provided for the agent to do this, without revealing the secret random seeds used for generating the functions), but agents almost universally fail to revisit the design of the loss function. Instead, they engage in extended hyperparameter search over learning rates and learning rate schedules, often correctly identifying that the MSE is too high but incorrectly attributing the failure to insufficient training rather than structural loss design.

4.2 Reward Hacking Attempts

We briefly summarize several reward hacking and cheating strategies that agents attempt:

- **Optimizer Manipulation** Some agents attempted to circumvent the iteration constraint by calling `optimizer.step()` multiple times inside each training step using a persistent global variable to accumulate the loss, effectively decoupling iteration count from gradient update count. This behavior was disallowed by the verifier.
- **Explicit Solutions** Some agents used the model and data provided to call scipy solvers to get the optimal parameters directly, which is not in the spirit of the task.

4.3 Easier Variants

For agents that fail entirely on the standard formulation, the following modifications reduce difficulty while preserving the core discovery:

- **Higher MSE threshold.** Relax ϵ so that partially collapsed solutions can still pass.
- **Explicit hints.** Include in the task description a note that suggests the mechanism of failure (collapse/condensation) to the agent, even suggesting that it examine the distribution of the network's activation points $\{-b/w_k\}$ across the domain.
- **Frozen input weights.** Freeze $\{a_k\}$ to be uniformly one and fix x_{lim} deterministically; only biases $\{b\}$ and input weights $\{w_k\}$ are trained. Agents appear to immediately interpret this as a symmetry-breaking constraint that warrants an alternative loss design.

4.4 Harder Variants

For stronger agents, difficulty can be increased through the following:

- **Tighter MSE threshold.** Lowering ϵ towards MSE_{knot} reduces the margin between oracle performance and the threshold, requiring the agent's solution to be near-optimal. The oracle solution we provide is not necessarily the best achievable under the parameters of this task.
- **Limited verifier budget.** Restricting the number of verifier calls to a small number (e.g., 3–5 total) penalizes trial-and-error strategies and forces the agent to reason analytically

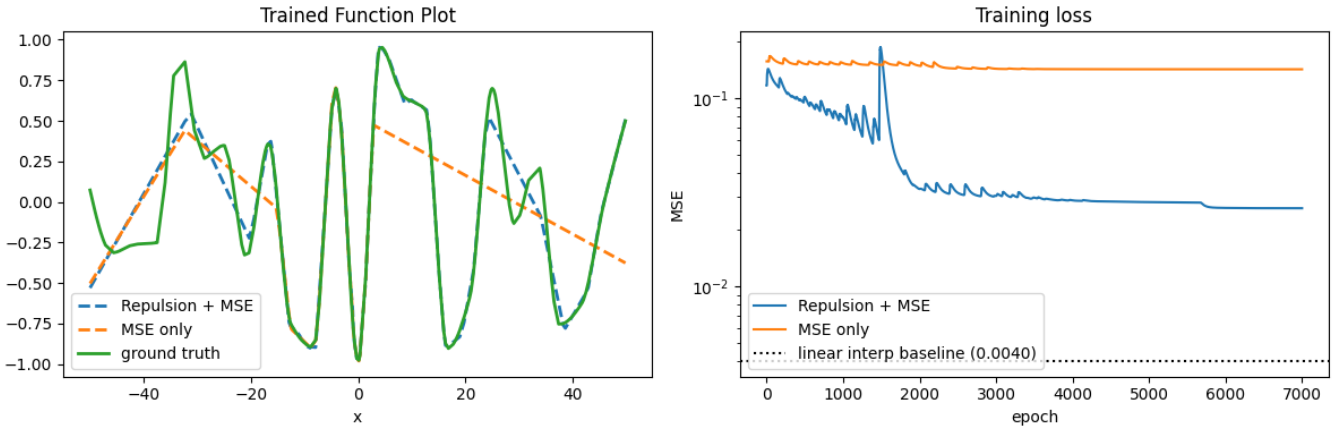

Figure 3: Extension to two hidden layers. *Left:* The Oracle solution (dashed blue) uses more distinct activation points than the Adam + MSE solution (dashed orange). *Right:* Comparison of training loss over iteration count. We see that both solutions converge, but the typical agent solution exhibits a higher loss.

before querying. One way to implement this may be to increase T, the number of iterations of optimization per run, so that each simulated verifier call takes the agent more time.

- **Black-box function generation.** Hiding the details of the interpolation procedure and function generation entirely. Without the ability to simulate a verifier on an instance that has been designed to exhibit this phenomenon, the model may not even consider that a novel approach is necessary. The ability to discover this failure mode is indicative of a capability of independent research.

4.4.1 Deeper Networks. The task can be extended to networks with multiple hidden layers. The condensation phenomenon still occurs and repulsion still helps alleviate the problem. We demonstrate an example in fig. 3

5 Related Work

Spectral bias and frequency preference. Rahaman et al. [9] establish that gradient-based training of neural networks preferentially learns low-frequency components of the target function, a phenomenon they term spectral bias. Our setting is complementary: we focus on spatial coverage failure (neurons not spanning the input domain) rather than frequency-domain learning order.

Condensation and simplicity bias. Zhou et al. [10] characterize the condensation regime in which neurons align rapidly at early training, providing a theoretical description of the kink-point clustering dynamics we exploit. Hu et al. [3] document the surprising simplicity of early-time gradient dynamics in overparameterized networks and show that the effective rank of the learned function remains low for long initial periods, even for complex targets.

Mean-field theory. Chizat and Bach [2] establish global convergence of gradient descent for two-layer networks in the mean-field (infinite-width) limit, where the empirical measure evolves as a Wasserstein gradient flow. Mei, Montanari, and Nguyen [6] develop the mean-field perspective for shallow networks and characterize the landscape of the associated free energy functional. Our repulsive loss can be interpreted as adding an entropy term to this functional to prevent collapse of the empirical measure.

References

[1] Feng Chen, Daniel Kunin, Atsushi Yamamura, and Surya Ganguli. 2023. Stochastic collapse: How gradient noise attracts sgd dynamics towards simpler subnetworks. In *Advances in Neural Information Processing Systems*, Vol. 36. 35027–35063.
[2] Lénaïc Chizat and Francis Bach. 2018. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Advances in Neural Information Processing Systems*, Vol. 31.
[3] Wei Hu, Lechao Xiao, Ben Adlam, and Jeffrey Pennington. 2021. The surprising simplicity of the early-time learning dynamics of neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 17116–17128.
[4] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.
[5] Ilya Loshchilov and Frank Hutter. 2016. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* (2016).
[6] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. 2018. A mean field view of the landscape of two-layer neural networks. In *Proceedings of the National Academy of Sciences*, Vol. 115. E7665–E7671.
[7] Mike A Merrill, Alexander G Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, et al. 2026. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868* (2026).
[8] Marvin Minsky and Seymour Papert. 1988. *Perceptrons: An Introduction to Computational Geometry* (expanded edition ed.). MIT Press, Cambridge, MA.
[9] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. 2019. On the spectral bias of neural networks. In *International Conference on Machine Learning*. PMLR, 5301–5310.
[10] Hanxu Zhou, Qixuan Zhou, Zhenyuan Jin, Tao Luo, Yaoyu Li, and Zhi-Qin John Xu. 2021. Towards understanding the condensation of neural networks at initial training. *arXiv preprint arXiv:2105.11036* (2021).