

UC Santa Barbara

UC Santa Barbara Electronic Theses and Dissertations

Title

Reuse or Overwrite: Training Dynamics of Rank-1 Recurrent Neural Networks

Permalink

<https://escholarship.org/uc/item/3979b8z7>

ISBN

9798186386711

Author

Ou, Tzu-Ming

Publication Date

2026-06-16

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

SANTA BARBARA

Reuse or Overwrite:

Training Dynamics of Rank-1 Recurrent Neural Networks

A Thesis submitted in partial satisfaction of the
requirements for the degree Master of Science
in Electrical and Computer Engineering

by

Tzu-Ming Ou

Committee in charge:

Professor Nina Miolane, Chair

Professor B.S. Manjunath

Professor Ramtin Pedarsani

June 2026

The thesis of Tzu-Ming Ou is approved.

Professor Ramtin Pedarsani

Professor B.S. Manjunath

Professor Nina Miolane, Committee Chair

June 2026

Abstract

Reuse or Overwrite: Training Dynamics of Rank-1 Recurrent Neural Networks

by

Tzu-Ming Ou

Recurrent neural networks (RNNs) are widely used both as practical architectures for machine learning tasks involving sequential data and as computational models that let neuroscientists study recurrent circuits in the brain, yet their training dynamics are only partially understood. A key reason is dimensionality: a small RNN can have thousands of parameters that interact nonlinearly through the recurrent dynamics, making the loss landscape difficult to analyse and the role of any single design choice difficult to isolate. This thesis takes a different approach. Rather than studying full-rank networks directly, we analyse a minimal model—the rank-1 RNN with connectivity $W = mn^\top$ trained on a one-bit flip-flop task. The training dynamics get reduced to a two-dimensional phase plane over effective connectivity parameters, allowing the network’s behaviour to be both derived mathematically and tested empirically.

Using a model reduction we show that the rank-1 RNN is characterised by two effective scalar parameters, the recurrent gain $R_{\text{eff}} = n^\top m$ and the input gain $M_{\text{eff}} = n^\top W_{\text{in}}$. We also derive a formula for the “flip boundary” $M_{\text{crit}}(R_{\text{eff}})$ that separates inputs strong enough to “flip-flop” the network from one memory state to the other when a new input pulse arrives. We then use this framework to study three questions about training. First, we show that the bias vector b , although not mathematically required for bistability, is an extremely helpful term during base training. Under standard small-weight initialisation, removing it caused gradient descent to have difficulty converging to the bistable attractor; we further show that strong-synapse initialisation breaks the same symmetry and reaches

the bistable target without any bias term, confirming that this failure is specific to the small-weight regime. We trace this failure to that the loss function treats positive and negative input gain identically and gradient descent sees no signal pushing it toward the correct solution from the start of training, as well as the fact that the readout direction n , which defines the $(R_{\text{eff}}, M_{\text{eff}})$ coordinate system, is itself free to rotate during training, so the optimiser cannot be relied upon to stay in the bistable region once found. After the base training, we show that retraining trajectories initialised above and below the flip boundary exhibit different behaviours. In agreement with the bifurcation prediction, below-boundary initialisations are more prone to “overwriting” existing bistable struction instead of “reusing” it, before recovering. Finally, we show that using separate learning rates for m and W_{in} produces faster and more direct trajectories in the $(R_{\text{eff}}, M_{\text{eff}})$ plane by allowing M_{eff} to climb away from the boundary before R_{eff} drifts.

These results give a quantitative, geometric account of when a network reuses an existing solution and when it must overwrite it, and provide hypotheses for future experiments on full-rank recurrent networks.

Contents

Abstract	iii
1 Introduction	1
1.1 The dimensionality obstacle	1
1.2 This thesis	2
1.3 Organisation	3
2 Related Work	4
2.1 Reverse-engineering trained recurrent networks	4
2.2 Low-rank recurrent neural networks	5
2.3 Multistability, initialisation, and the difficulty of training	6
2.4 Dynamical motifs and reuse across tasks	7
2.5 Position of the present work	7
3 Preliminaries	9
3.1 Discrete-time leaky-tanh RNN	9
3.2 Rank-1 parameterisation	9
3.3 The one-bit flip-flop task	10
3.4 The cubic normal form	11
4 Methods	13
4.1 Scalar reduction of the rank-1 dynamics	13
4.2 Model reduction by linearisation	13
4.3 Bifurcation and flip-boundary analysis	14
4.4 Role of the bias vector	16
4.5 Gradient projection for two-dimensional trajectories	17
4.6 Training and retraining protocols	18

5	Results	20
5.1	Base training: bias free versus bias frozen	20
5.2	Initialisation experiment: is a good start enough?	22
5.3	Strong-synapse initialisation removes the need for bias	24
5.4	Retraining experiments: above and below the flip boundary	25
5.5	Short summary of empirical findings	30
6	Discussion	33
6.1	Bias as a symmetry-breaker	33
6.2	The moving coordinate-frame problem	33
6.3	What the rank-1 analysis is good for	34
6.4	Limitations and future directions	34
7	Conclusion	36
	References	38

List of Figures

- 1 One-bit flip-flop task. Top: sparse input pulses ($p = 0.1$); blue = +1, red = -1, zero otherwise. Bottom: target output, held at the sign of the most recent pulse until the next one arrives. 11

- 2 Base training with b free (blue) versus $b = 0$ frozen (red). Circles mark the start and squares the end of each trajectory. **(a)** Gradient field in $(R_{\text{eff}}, M_{\text{eff}})$ space: the magenta curve is the flip boundary $M_{\text{crit}}(R_{\text{eff}})$; the shaded red region is the monostable regime ($R_{\text{eff}} < 1$). With b free the trajectory enters the bistable region and converges to $(R_{\text{eff}}, M_{\text{eff}}) \approx (1.50, 7.60)$; with $b = 0$ frozen it reaches $R_{\text{eff}} = 1.5$ before drifting to a monostable solution. **(b)** Log-scale loss landscape with the same trajectories overlaid. **(c)** Training loss versus epoch for both conditions. Both endpoints achieve comparable MSE (≈ 0.05), illustrating that training loss alone does not distinguish bistable from monostable solutions. 21

- 3 Base training from four initialisations with $b = 0$ frozen: **(a)** zero ($R_{\text{eff}} = 0$, $M_{\text{eff}} = 0$), small ($R_{\text{eff}} = 0.1$, $M_{\text{eff}} = 0.1$), normal ($R_{\text{eff}} = 1.0$, $M_{\text{eff}} = 1.0$), and bistable ($R_{\text{eff}} = 1.1$, above $M_{\text{crit}}(1.1)$). Even the bistable initialization cannot stay in bistable reigon because n rotates freely during base training and no gradient projection is applied, the optimiser finds a non-bistable solution along a different axis. The phase-portrait trajectories here are measured in a *moving* coordinate frame and are therefore not directly comparable to the fixed-frame retraining trajectories of Experiments 1–3 in the following subsection. **(b)** All runs achieve low MSE loss. 23

- 4 Standard ($n \sim \mathcal{N}(0, 1/N)$, red) versus strong-synapse ($n \sim \mathcal{N}(0, 1)$, blue) initialisation with $b = 0$ frozen. **(a)** phase portrait; circles mark initialisation, squares convergence. Strong init reaches higher R_{eff} at lower M_{eff} . **(b)** loss curves; strong init converges more slowly but reaches comparable loss. 26

5	Experiment 1: b free during retraining, no b_{eff} correction. (a)(c)(e) gradient field with the flip boundary (magenta) and fold curves (dashed black). (b)(d)(f) log-scale loss landscape with the same overlays. Trajectories exhibit visible zigzagging near convergence as b_{eff} drifts throughout optimisation, continuously shifting the effective flip boundary.	28
6	Experiment 2: corrected baseline with $b_{\text{eff}} = 0$ and b frozen. Above-boundary initialisations climb smoothly through the bistable region; below-boundary initialisations at $R_{\text{eff}} = 1.3$ transiently lose bistability before recovering.	29
7	Experiment 3: separate learning rates for m and W_{in} . Trajectories reach target R_{eff} values significantly faster in the first 30 steps compared to Experiment 2 (see Table 3), and curve less toward the bifurcation $R_{\text{eff}} = 1$, yielding more direct paths in the $(R_{\text{eff}}, M_{\text{eff}})$ plane. The final asymptotic location is essentially the same as in Experiment 2, indicating that the ceiling at $M_{\text{eff}} \approx 6$ is a property of the projected loss landscape rather than an optimiser artefact.	31

List of Tables

1	Base training with $b = 0$ frozen, 5 seeds $\times$ 4 learning rates (6 000 epochs each, Adam). Only $\text{lr}=1\text{e-}3$ reaches $R_{\text{eff}} > 1$, but barely ($R_{\text{eff}} \in [1.01, 1.10]$ versus the target $R_{\text{eff}} \approx 1.50$ reached with b free); all other settings find the monostable high-gain solution or diverge.	22
2	Base training with $b = 0$ frozen under standard versus strong-synapse initialisation (Adam, $\text{lr} = 10^{-3}$, 6 000 epochs). Standard init reaches only $R_{\text{eff}} \approx 1.1$; strong-synapse init reaches $R_{\text{eff}} \approx 1.5$ in every seed without any bias term.	25
3	Step-by-step comparison for the $R_{\text{eff}} = 1.3$, above-boundary trajectory. At step 1 both experiments share the same R_{eff} drop ($1.300 \rightarrow 1.081$), but M_{eff} jumps to 3.35 in Experiment 3 versus 2.47 in Experiment 2 because the learning rate on W_{in} is five times larger. By step 5, Experiment 3 has reached $R_{\text{eff}} = 1.387$ with loss 0.19 while Experiment 2 is still at $R_{\text{eff}} = 1.006$ with loss 0.93. Both converge to the same ceiling by step 500.	32

1 Introduction

RNNs are commonly used for machine learning and neuroscience research [Sussillo and Barak, 2013, Mante et al., 2013, Vyas et al., 2020, Barak, 2017, Yang and Wang, 2020]. Both communities have invested heavily in training RNNs on field specific tasks but the trained networks have often been treated as black boxes. Even on simple benchmarks, why a particular training run succeeds or fails and which features of the trained network are generic versus accidental is not fully explored.

1.1 The dimensionality obstacle

An example small RNN with 100 hidden units that is randomly initialised, it has $100^2 = 10,000$ parameters in its recurrent matrix alone. Additionally, the loss landscape over those parameters is high-dimensional, non-convex, and shaped by recurrent dynamics in ways that are difficult to summarise. To tackle this dimensionality obstacle, two broad strategies have emerged in response. The first is to train an RNN and then reverse-engineer through fixed-point analysis and local linearisation [Sussillo and Barak, 2013, Maheswaranathan et al., 2019a,b, Yang et al., 2019, Driscoll et al., 2024]. The second is to constrain the network’s structure during training through low-rank parameterisations $W = \sum_{r=1}^R m^{(r)} n^{(r)\top}$, so that the dynamics become analytically tractable from the start [Mastrogiuseppe and Ostojic, 2018, Schuessler et al., 2020, Beiran et al., 2021, Dubreuil et al., 2022]. The low-rank approach has the additional appeal because gradient descent on a full-rank network appears to add low-rank corrections to the initial connectivity [Schuessler et al., 2020], suggesting that the low-rank is not merely a convenient constraint but a description of what training tends to produce.

Even within this low-rank model, there is a gap between the theoretical structure of a trained network and what the step-by-step training dynamic looks like. Questions about the geometry of optimisation such as Why does removing a single bias term complicate

training on a task whose fixed-point structure is unchanged by that removal? Or Why does initialising inside the bistable region not guarantee that training keeps it there? These are questions difficult to answer in a full-rank network because the parameters relevant to the answer are not directly observable in the trained weights, making investigating low-rank networks meaningful.

1.2 This thesis

This thesis takes the simplest non-trivial member of the low-rank family: the rank-1 RNN with connectivity $W = mn^\top$, trained on a one-bit flip-flop task. The flip-flop task is a standard benchmark in the study of RNN dynamics [Sussillo and Barak, 2013]: the network sees a stream of inputs that pulse between ± 1 and must hold the most recent pulse on its output until the next pulse arrives. The natural dynamical implementation is bistability with two stable fixed points encoding the two memory states and a flip boundary in input space separating pulses that succeed in switching the state from those that do not.

We will show the rank-1 RNN quantitatively with a model reduction collapsing the N -dimensional dynamics onto a single scalar variable governed by two effective parameters: the recurrent gain $R_{\text{eff}} = n^\top m$ and the input gain $M_{\text{eff}} = n^\top W_{\text{in}}$. Training trajectories can be plotted as paths in the $(R_{\text{eff}}, M_{\text{eff}})$ plane on top of the loss landscape and the flip boundary, making the geometry of optimisation directly visible.

We use this framework to address three questions. *First*, what is the role of the bias vector b during training? Bias is not required for bistability yet networks trained with b frozen at zero consistently fail to stay in the bistable regime if entered. We show via a symmetry argument that this failure is structural where without b , the loss is symmetric in M_{eff} at initialisation, so gradient descent sees no signal in the direction it must move and becomes trapped in a monostable solution. *Second*, do retraining trajectories starting on either side of the flip boundary behave differently, as the bifurcation analysis predicts?

We initialise networks at three values of R_{eff} , each above and below the flip boundary and confirm that below-boundary initialisations lose bistability more easily and takes longer to recover. *Third*, can the geometry of the $(R_{\text{eff}}, M_{\text{eff}})$ plane be exploited to accelerate training? We show that giving W_{in} a higher learning rate than m produces faster convergence by allowing M_{eff} to cross the boundary before R_{eff} drifts toward the monostable regime.

To summarize, the main contributions of this thesis are: a model reduction that transforms N-dimensional activities into a single scalar variable and a closed-form expression for the flip boundary $M_{\text{crit}}(R_{\text{eff}})$; an analysis showing that bias is essential during base training and that its absence cannot be compensated by repositioning at fixed weight scale, but can be compensated by rescaling the initialisation, because the readout vector n rotates freely and takes the network off the analysed axis; an empirical validation of the flip-boundary prediction across three values of R_{eff} ; and retraining strategies that improve performance. This narrow experiment design is intentional, with the simplest possible model, each question becomes answerable by hand and the results can serve as a concrete baseline for the same questions in larger networks.

1.3 Organisation

Section 2 reviews the relevant literature on RNNs. Section 3 defines the rank-1 RNN, the flip-flop task, and the cubic normal form that motivates the analysis. Section 4 carries out the model reduction, derives the flip boundary, discusses the role of bias, and describes the retraining protocols including the effect of freezing n on parameter trajectories in the $(R_{\text{eff}}, M_{\text{eff}})$ plane. Section 5 presents the base-training comparison, the initialisation experiments, and the three retraining strategies. Section 6 interprets these results and outlines how the framework might extend to higher-rank networks and other tasks. Section 7 concludes.

2 Related Work

This thesis touches on four trends in the literature on recurrent neural networks: reverse-engineering trained networks through fixed-point analysis, the theory of low-rank RNNs, the relationship between multistability and learning, and the identification of reusable dynamical motifs. We review each in turn, with an emphasis on commonalities with the rank-1 analysis developed in Sections 4 and 5.

2.1 Reverse-engineering trained recurrent networks

A good starting point for modern dynamical-systems analyses of RNNs is Sussillo and Barak [2013], who showed that a trained network’s behaviour can often be understood by locating the fixed and slow points of its dynamics and linearising around them. They introduced a simple optimisation procedure for finding such points and applied it to networks trained on several tasks, including the three-bit flip-flop. They found that even networks with hundreds of units organise themselves around a small number of fixed points, establishing fixed-point analysis as the first step in interpreting a trained RNN.

Maheswaranathan et al. [2019a] applied the same methodology to multiple RNN architectures trained on the same task and found they all converge to a similar low-dimensional solution, suggesting that the underlying computational mechanism is shared across architectures rather than specific to any one design. This motivates studying the simplest reproducible model. The rank-1 RNN studied here is one such model, its dynamics are governed by scalar parameters $R_{\text{eff}} = n^\top m$ and $M_{\text{eff}} = n^\top W_{\text{in}}$. They can be computed directly from the weight matrices during training.

A complementary infrastructure contribution is the Computation-through-Dynamics Benchmark of Versteeg et al. [2025], which provides standardised synthetic datasets with known ground-truth dynamics together with quantitative metrics for evaluating how well data-driven models recover the underlying computation. Their benchmark includes

the three-bit flip-flop as an entry-level task, chosen precisely because its dynamics are simple and well-characterised. The one-bit flip-flop we study is the minimal version of this same family. From the training side, Haputhanthri et al. [2025] show that the phase-space structure of an RNN reorganises in discrete steps over training where abrupt drops in loss coincide with sudden geometric restructuring of the dynamics and introduce a temporal-consistency regulariser that encourages these structures to form, accelerating training in the strongly connected regime. Their work is direct evidence that training dynamics cannot be read off from the final fixed-point structure alone, which motivates tracking $(R_{\text{eff}}, M_{\text{eff}})$ over the course of training rather than only at convergence.

The visualisation of training dynamics has also been pursued from a data-driven direction. Gigante et al. [2019] embed a network’s hidden representations across training epochs into a two-dimensional diffusion space (M-PHATE), revealing patterns such as catastrophic forgetting and heterogeneity-driven generalisation that are invisible from loss curves alone. Xie [2024] adapts this machinery to recurrent networks and shows that similar geometric signatures track learning in RNNs.

2.2 Low-rank recurrent neural networks

Mastrogiuseppe and Ostojic [2018] introduced the theoretical framework this thesis builds on. They showed that RNNs with low-rank connectivity structure have low-dimensional dynamics that can be described by a small number of scalar variables derived directly from the connectivity. In their framework, a single scalar $\kappa = \mathbf{n}^\top \mathbf{r}$ captures the entire low-rank dynamics, with bistability emerging when the recurrent gain exceeds a critical threshold, this connection is shown in Dinc et al. [2025]. This thesis works in the simpler setting where the random connectivity component is removed entirely, leaving a purely rank-1 network. The model reduction in Section 4.2 is a discrete-time adaptation of this framework to the leaky-tanh RNN used here.

Schuessler et al. [2020] provide an important complement, with related evidence

in Krause et al. [2022]. Training RNNs on neuroscience-inspired tasks including the flip-flop task, they observe that the weight change $\Delta W = W - W_0$ where W_0 is the initial connectivity matrix and W is the connectivity after training, is induced by gradient descent and is approximately low-rank even when training operates in the full-rank parameter space. Their analysis shows this is a consequence of the low-dimensional task structure, and that the random initial connectivity W_0 accelerates learning by amplifying the effect of low-rank gradient updates through correlations between W_0 and ΔW . This result has two implications for the present work. First, it suggests the rank-1 RNN is not merely a toy model, it is the minimal version of what training tends to produce on low-dimensional tasks. Second, it motivates the distinction between W_0 and ΔW that underlies the bias analysis in Section 4.4, where we show that bias acts during base training in a way that cannot be reproduced by clever initialisation alone.

2.3 Multistability, initialisation, and the difficulty of training

A reference for the difficulty of training RNNs on long-memory tasks is Pascanu et al. [2013], who diagnose vanishing and exploding gradients from analytical, geometric, and dynamical-systems perspectives. They show that gradient explosions correspond to cliff-like structures in the loss surface that arise when the network crosses boundaries between basins of attraction and that bifurcation boundaries in parameter space are one mechanism by which such crossings occur.

Lambrechts et al. [2023] take this idea further and ask whether training can be made easier by initialising networks already in the multistable regime. They introduce a warmup procedure that maximises a differentiable proxy for the number of reachable attractors at initialisation, and show that on a range of long-memory benchmarks warmed-up networks learn substantially faster than randomly initialised ones. They also show that several existing initialisation tricks in the literature implicitly achieve the same effect. This is closely related to the “initialisation cannot substitute for bias” result in Section 4.4,

but with an important distinction: Lambrechts et al. [2023] show that initialising in the multistable regime helps when all parameters are subsequently free, including the bias. We instead show that under a bias-frozen protocol, even an initialisation inside the bistable regime is not preserved by gradient descent.

2.4 Dynamical motifs and reuse across tasks

A core question for this thesis is whether the parameter-space picture we develop can be reused when the network’s parameters are perturbed. The cleanest articulation comes from Driscoll et al. [2024], who train networks on a range of interrelated tasks and find that the network organises its solution around a small set of dynamical motifs, including fixed point attractors, decision boundaries, rotational structures, that are reused across tasks. When a network learns a new task, it reuses the same underlying dynamical structures from previous tasks where possible. This reframes the question of what changes during retraining into which motifs are kept and which are reorganised.

The rank-1 network has a single dynamical motif: two stable memory states separated by an unstable boundary. Our retraining experiments test whether this motif survives when the network’s parameters are perturbed. Starting above the flip boundary means the motif is still intact and retraining only fine-tunes it; starting below means the motif has been lost and must be rebuilt. This is the simplest possible test of motif preservation in the sense of Driscoll et al. [2024].

2.5 Position of the present work

In summary, trained RNNs are increasingly interpretable through fixed-point analysis and low-rank theory, and their learned dynamical motifs are recognised as reusable computational structures whose emergence depends critically on initialisation and training dynamics. What is largely missing is a tight, fully analytical account of how these pieces fit together during training on one task in one model at the level of individual parameter

trajectories. The work in this thesis is an attempt to provide such an account for the rank-1 RNN on the flip-flop task.

3 Preliminaries

This section reviews the three pieces of background needed for the analysis in Sections 4–5: the discrete-time leaky-tanh RNN and its rank-1 specialisation, the one-bit flip-flop task that defines the training objective, and the one-dimensional cubic normal form whose bifurcation structure the rank-1 RNN inherits.

3.1 Discrete-time leaky-tanh RNN

Let $r[t] \in \mathbb{R}^N$ denote the hidden state of an N -unit RNN at discrete time t , and let $u[t] \in \mathbb{R}$ denote the scalar input. We use the leaky-tanh update

$$r[t+1] = (1 - \alpha) r[t] + \alpha \tanh(W_{\text{rec}} r[t] + b + W_{\text{in}} u[t]), \quad (1)$$

with recurrent matrix $W_{\text{rec}} \in \mathbb{R}^{N \times N}$, input weight vector $W_{\text{in}} \in \mathbb{R}^{N \times 1}$, bias vector $b \in \mathbb{R}^N$, and leak rate $\alpha \in (0, 1]$. Throughout the thesis we set $\alpha = 0.5$ and $N = 100$ unless stated otherwise. The scalar output is defined via the right connectivity vector n (introduced in Section 3.2):

$$o[t] = \kappa[t] := n^\top r[t], \quad (2)$$

where $\kappa[t]$ also serves as the collective variable in the model reduction of Section 4.2. The initial hidden state at the start of each trial is drawn from $\mathcal{N}(0, 1/9)$ rather than set to zero, to avoid initialising at the fixed point $r = 0$ when $b = 0$ and $u = 0$.

3.2 Rank-1 parameterisation

The rank-1 model is the specific case of Eq. (1) in which W_{rec} is constrained to be the outer product of two learned vectors:

$$W_{\text{rec}} = m n^\top, \quad m, n \in \mathbb{R}^N. \quad (3)$$

where $m, n \in \mathbb{R}^N$ are initialised entrywise from $\mathcal{N}(0, 1/N)$ while the bias b initialized as zero, so that the entries of W_{rec} are of order $1/N$, consistent with the normalisation convention of Mastrogiuseppe and Ostojic [2018]. Substituting Eq. (3) into Eq. (1) gives

$$r[t+1] = (1 - \alpha) r[t] + \alpha \tanh(m n^\top r[t] + b + W_{\text{in}} u[t]). \quad (4)$$

Multiplying both sides on the left by $n^\top$ and defining $\kappa[t] := n^\top r[t]$ yields the exact scalar equation

$$\kappa[t+1] = (1 - \alpha) \kappa[t] + \alpha n^\top \tanh(m \kappa[t] + b + W_{\text{in}} u[t]), \quad (5)$$

which is exact for any rank-1 network. Section 4.2 introduces the model reduction that closes this equation in terms of κ alone.

3.3 The one-bit flip-flop task

The flip-flop task is the benchmark used in Sussillo and Barak [2013] and many follow-up works. We use a one-bit version where at each time step t , with probability $p = 0.1$ the input $u[t]$ pulses to ± 1 (sign uniform), otherwise $u[t] = 0$; the target output is the sign of the most recent non-zero pulse, held constant until the next pulse. A successful network must therefore implement two long-lived memory states corresponding to “the last pulse was $+1$ ” and “the last pulse was -1 ”, and must be able to transition between them when the next pulse arrives.

For training we generate $B = 64$ independent trials of length $T = 200$ time steps from this distribution and use the mean squared error between the network output $o[t]$ and the target as the loss. The choice of p and T ensures that the average run contains many pulse events and many memory-hold segments per trial. Inputs are held fixed across all retraining experiments to remove sample variability as a source of trajectory differences in the $(R_{\text{eff}}, M_{\text{eff}})$ plane.

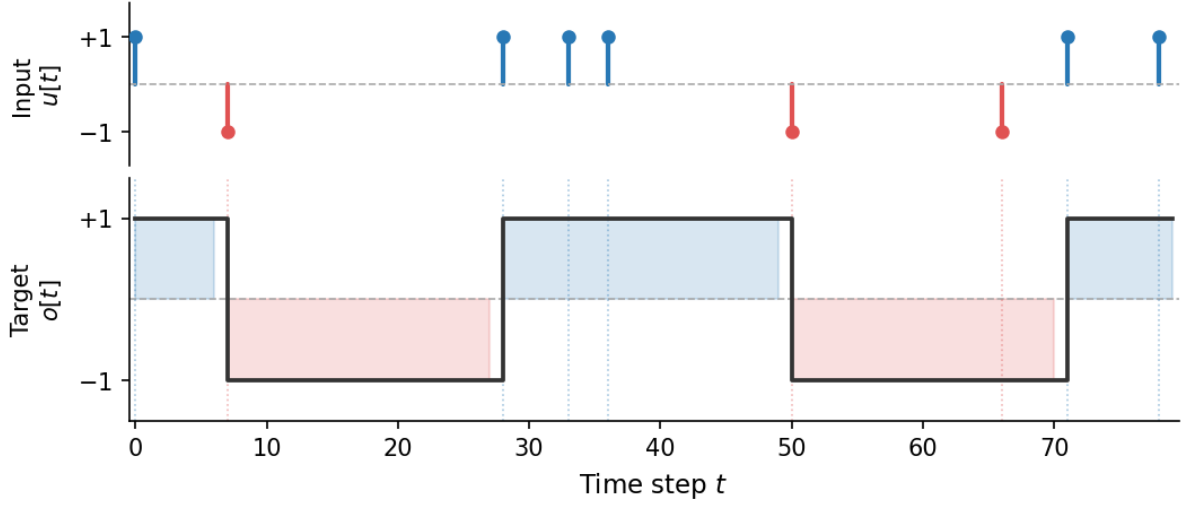


Figure 1: One-bit flip-flop task. Top: sparse input pulses ($p = 0.1$); blue = +1, red = -1, zero otherwise. Bottom: target output, held at the sign of the most recent pulse until the next one arrives.

3.4 The cubic normal form

A network that solves the flip-flop task needs to maintain two distinct stable states and switch between them when an input pulse arrives. The simplest mathematical model with this structure is the cubic normal form

$$\dot{x} = -x^3 + r x + \mu u(t), \quad (6)$$

with recurrent gain r and input gain μ . When $r > 0$ the system is bistable with stable fixed points at $x^* = \pm\sqrt{r}$; when $r < 0$ it is monostable. The boundary between flip-capable and flip-incapable parameter regions, the flip boundary, occurs at

$$M_{\text{crit}}(r) = \frac{\sqrt{r}}{\tau_{\text{flip}}}, \quad (7)$$

where τ_{flip} is the pulse duration. For a unit-duration pulse this reduces to $M_{\text{crit}}(r) = \sqrt{r}$. The parameter plane (r, μ) can be divided into three regions: monostable ($r < 0$), bistable but unable to switch on a single pulse ($r > 0$, $\mu < M_{\text{crit}}$), and bistable and flip-capable

$(r > 0, \mu > M_{\text{crit}})$ which is the preferred region to solve the flip-flop task.

We introduce this system as a toy model. The rank-1 RNN inherits the same pitchfork bifurcation structure and a motivationally similar flip boundary, both derived in Section 4.3.

4 Methods

This section develops the model reduction of the rank-1 RNN, derives its flip boundary in closed form, analyses the role of the bias vector during training, and describes the gradient-projection scheme that confines retraining to the two-dimensional effective parameter plane. Together they define the analytical framework for Section 5.

4.1 Scalar reduction of the rank-1 dynamics

We begin from the scalar dynamics derived in Section 3.2. Substituting the rank-1 factorisation $W_{\text{rec}} = m n^\top$ into the leaky-tanh update and using the projection $\kappa[t] = n^\top r[t]$ yields

$$\kappa[t+1] = (1 - \alpha) \kappa[t] + \alpha n^\top \tanh(m \kappa[t] + b + W_{\text{in}} u[t]). \quad (8)$$

This stands for any rank-1 network.

4.2 Model reduction by linearisation

The two scalar parameters that govern the reduced dynamics emerge exactly from linearising Eq. (5) about the origin. For small κ and u , $\tanh(x) \approx x$, so

$$n^\top \tanh(m \kappa + W_{\text{in}} u) \approx (n^\top m) \kappa + (n^\top W_{\text{in}}) u = R_{\text{eff}} \kappa + M_{\text{eff}} u, \quad (9)$$

which defines

$$R_{\text{eff}} = n^\top m, \quad M_{\text{eff}} = n^\top W_{\text{in}}. \quad (10)$$

The recurrent gain R_{eff} is the single non-zero eigenvalue of the rank-one matrix $W_{\text{rec}} = m n^\top$ (with eigenvector m), and it sets the linear stability of the $\kappa = 0$ fixed point: the origin is stable for $R_{\text{eff}} < 1$ and loses stability at $R_{\text{eff}} = 1$, giving a pitchfork bifurcation. This location depends only on the slope of the nonlinearity at the origin, not its detailed form. The input gain M_{eff} sets how strongly a pulse displaces the state.

The linearisation in Eq. (9) is exact only near the origin. To study the bistable regime we need a nonlinear closure, and we adopt the minimal one for reproducing the exact linearisation at the origin and replacing the projected nonlinearity by a scalar $\tanh$ with the same slope and input sensitivity,

$$\kappa[t+1] = (1 - \alpha) \kappa[t] + \alpha \tanh(R_{\text{eff}} \kappa[t] + M_{\text{eff}} u[t]). \quad (11)$$

Eq. (11) is an approximation to get a reduced model. The exact right-hand side, $n^\top \tanh(m\kappa + W_{\text{in}}u)$, depends on the full joint distribution of $\{m_i, n_i, (W_{\text{in}})_i\}$ and does not reduce to a single $\tanh$ in general. It is exact at the level of the linearisation and the bifurcation point; its predictions in the nonlinear regime, including the flip boundary derived next, are validated empirically against the full N -dimensional network in Section 5.

4.3 Bifurcation and flip-boundary analysis

During a memory-retention phase ($u = 0$), the fixed points of the reduced model in Eq. (11) satisfy

$$\kappa^* = \tanh(R_{\text{eff}} \kappa^*). \quad (12)$$

This is exact for the reduced model but approximate for the full network, since Eq. (11) itself is an approximation; the true fixed points solve $\kappa^* = \sum_i n_i \tanh(m_i \kappa^*)$, which we do not attempt to solve in closed form. For $R_{\text{eff}} \leq 1$ the only solution is $\kappa^* = 0$; for $R_{\text{eff}} > 1$ there are three, $\kappa^* \in \{-\kappa_+, 0, +\kappa_+\}$, with $\pm\kappa_+$ stable and 0 unstable. This is the pitchfork inherited from the cubic system of Section 3.4, with the critical parameter at $R_{\text{eff}} = 1$.

To switch memory states, a flip pulse must carry the state from a stable fixed point across the basin boundary at $\kappa = 0$. Following the toy model, we estimate the displacement produced by a single pulse from the instantaneous velocity at the fixed point. Suppose the network sits at $\kappa^* = +\kappa_+$ and a flip pulse $u = -1$ is applied. The velocity of the

reduced model at that point is

$$\dot{\kappa}\Big|_{\kappa_+, u=-1} = -\kappa_+ + \tanh(R_{\text{eff}} \kappa_+ - M_{\text{eff}}), \quad (13)$$

and the displacement accumulated over the pulse duration τ_{flip} is $\Delta\kappa \approx \dot{\kappa} \tau_{\text{flip}}$, where $\tau_{\text{flip}} = \alpha$ for a single-step pulse. The flip succeeds when this displacement is large enough in magnitude to carry the state to or past the origin, $|\Delta\kappa| \geq \kappa_+$. The critical input gain M_{crit} is the value at which the displacement lands the state exactly on the boundary:

$$\kappa_+ + \alpha \left[\tanh(R_{\text{eff}} \kappa_+ - M_{\text{crit}}) - \kappa_+ \right] = 0. \quad (14)$$

Solving for M_{crit} gives

$$\tanh(R_{\text{eff}} \kappa_+ - M_{\text{crit}}) = -\frac{1-\alpha}{\alpha} \kappa_+ \implies M_{\text{crit}} = R_{\text{eff}} \kappa_+ + \text{arctanh}\left(\frac{1-\alpha}{\alpha} \kappa_+\right), \quad (15)$$

and using the fixed-point relation $R_{\text{eff}} \kappa_+ = \text{arctanh}(\kappa_+)$,

$$M_{\text{crit}}(R_{\text{eff}}) = \text{arctanh}(\kappa_+) + \text{arctanh}\left(\frac{1-\alpha}{\alpha} \kappa_+\right). \quad (16)$$

For $\alpha = 0.5$ the ratio $(1-\alpha)/\alpha = 1$, so this simplifies to $M_{\text{crit}}(R_{\text{eff}}) = 2 \text{arctanh}(\kappa_+)$, where $\kappa_+ = \kappa_+(R_{\text{eff}})$ is the positive fixed point from Eq. (12). The arctanh terms are well-defined since $\kappa_+ \in (-1, 1)$ and $(1-\alpha)/\alpha \leq 1$ for $\alpha \geq 0.5$.

Initialisations with $M_{\text{eff}} > M_{\text{crit}}$ flip in a single step; those with $M_{\text{eff}} < M_{\text{crit}}$ cannot, and at the sparse input rate $p = 0.1$ multi-step accumulation is unlikely to drive a transition reliably. Because Eq. (16) is derived from the reduced model rather than the full network, it is a prediction to be tested: we treat $M_{\text{crit}}(R_{\text{eff}})$ as an approximate boundary and validate it empirically in Section 5, where it is overlaid on every retraining figure.

4.4 Role of the bias vector

The bias $b \in \mathbb{R}^N$ contributes an effective scalar offset

$$b_{\text{eff}} = n^\top b, \quad (17)$$

which shifts the reduced model fixed-point equation to $\kappa^* = \tanh(R_{\text{eff}} \kappa^* + b_{\text{eff}})$, displacing both fixed points asymmetrically and shifting the flip boundary by b_{eff} .

Although $b_{\text{eff}} = 0$ fully supports bistability for any $R_{\text{eff}} > 1$, and our retraining experiments confirm the network operates stably there when initialised with $b = 0$ (Section 5.4), the role of bias during base training is different. Without bias the loss satisfies

$$L(R_{\text{eff}}, M_{\text{eff}}) = L(R_{\text{eff}}, -M_{\text{eff}}). \quad (18)$$

This follows from the joint sign flip $(m, n) \rightarrow (-m, -n)$, which leaves $R_{\text{eff}} = n^\top m$ unchanged while sending $M_{\text{eff}} = n^\top W_{\text{in}} \rightarrow -M_{\text{eff}}$, $\kappa = n^\top r \rightarrow -\kappa$, and $b_{\text{eff}} = n^\top b \rightarrow -b_{\text{eff}}$. Equivalently, in the reduced model the map $\kappa \rightarrow -\kappa$ sends $M_{\text{eff}} \rightarrow -M_{\text{eff}}$ at fixed R_{eff} because $\tanh$ is odd. Since the flip-flop target is also odd under a global sign flip of input and output, the loss is invariant under this transformation provided $b_{\text{eff}} = 0$, therefore it is even in M_{eff} . Also, this forces $\partial L / \partial M_{\text{eff}} = 0$ at $M_{\text{eff}} = 0$ and random initialisation places the network near $M_{\text{eff}} = 0$ in expectation, with fluctuations of order $1/\sqrt{N}$; close to this symmetric point gradient descent sees little signal in the M_{eff} direction and instead finds a monostable high-gain solution with comparable loss and a larger basin of attraction. With b free, b_{eff} breaks this symmetry and generates a gradient that steers the network toward the bistable basin. The empirical evidence is in Section 5.1.

One might ask whether repositioning the network at fixed weight scale could substitute for bias. The answer is no because n is free during base training, the coordinates $R_{\text{eff}} = n^\top m$ and $M_{\text{eff}} = n^\top W_{\text{in}}$ are measured along a moving axis. Gradient descent is therefore free to rotate n and find a solution along a different axis entirely, which is

exactly what it does. Even starting at $(R_{\text{eff}}, M_{\text{eff}}) = (1.1, 1.41)$, squarely above the flip boundary, the network collapses to a monostable solution at convergence (Section 5.2). Bias is needed not to reach the bistable regime but to keep gradient descent on the correct axis from the start of training.

However, bias is not the only way to break this symmetry. Rescaling the readout vector to $n \sim \mathcal{N}(0, 1)$ gives $\text{Var}(R_{\text{eff}}) = \text{Var}(M_{\text{eff}}) = 1$, so both effective parameters begin at $O(1)$ and gradient descent starts from a point with a meaningful M_{eff} gradient. We confirm in Section 5.3 that strong-synapse initialisation reaches the bistable target with $b = 0$ frozen, exactly as the symmetry argument predicts. Bias is therefore the symmetry-breaker required in the small-weight regime, not a general requirement for bistable training.

After base training, we project out the n -component of b to enforce $b_{\text{eff}} = 0$ before retraining:

$$b \leftarrow b - \frac{n^\top b}{\|n\|^2} n \quad \implies \quad n^\top b = 0. \quad (19)$$

This ensures the flip-boundary formula (16) remains valid throughout retraining. The bias is then frozen. We refer to this network as the corrected baseline; the uncorrected base model is retained separately for Experiment 1 in Section 5.4.

4.5 Gradient projection for two-dimensional trajectories

Optimising parameters m and W_{in} produces trajectories in $(R_{\text{eff}}, M_{\text{eff}})$ space that are noisy and hard to interpret because gradient steps scatter across all N directions in the network rather than moving cleanly along the effective coordinates. To reduce the system to only two degrees of freedom we project every gradient onto the readout direction before each optimiser step. We define $\hat{n} = n/\|n\|^2$ rather than the unit vector $n/\|n\|$ so that $n^\top(c\hat{n}) = c$ for any scalar c , which ensures that adding $\Delta R_{\text{eff}} \cdot \hat{n}$ to m shifts $R_{\text{eff}} = n^\top m$

by exactly ΔR_{eff} , and similarly for M_{eff} . The projection is then

$$\nabla_p L \leftarrow (\nabla_p L \cdot \hat{n}) \hat{n}, \quad p \in \{m, W_{\text{in}}\}. \quad (20)$$

This forces $\Delta R_{\text{eff}} = n^\top \Delta m$ and $\Delta M_{\text{eff}} = n^\top \Delta W_{\text{in}}$ to be the only free updates, yielding smooth trajectories directly comparable to the one-dimensional cubic system. The vector n is fixed after base training so that the coordinate system defined by $\hat{n}$ does not rotate during optimisation, keeping $(R_{\text{eff}}, M_{\text{eff}})$ geometrically consistent throughout. We use RMSprop without momentum, which handles the asymmetric $R_{\text{eff}}/M_{\text{eff}}$ loss landscape without distortion from accumulated momentum. This projection constrains training to the two effective degrees of freedom $(R_{\text{eff}}, M_{\text{eff}})$, which makes trajectories directly comparable to the reduced model but does not capture how these quantities evolve under unconstrained gradient descent; tracking $(R_{\text{eff}}, M_{\text{eff}})$ as observables during ordinary training is left to future work.

4.6 Training and retraining protocols

The full Rank-1 experiment consists of two phases: base training from random initialisation, and retraining from a perturbed copy of the base network.

For base training, parameters are initialised as in Section 3.1 and trained with Adam at learning rate 10^{-2} for 3 000 epochs (b -free condition). For the $b = 0$ frozen comparison we use Adam at 3×10^{-3} for 6 000 epochs; the smaller learning rate compensates for the lack of bias flexibility and the longer schedule rules out under-training. Each epoch is one pass over the $B = 64$ training trials with mean squared error loss. Adam is used here because its momentum accelerates convergence from a random initialisation, where reliability matters more than trajectory interpretability.

For retraining, we use the trained network but apply a parameter perturbation along $\hat{n}$, and retrain for 2000 RMSprop steps with gradient projection as in (20). RMSprop is

used instead of Adam because momentum would carry inertia from past gradient steps into the projected trajectory, obscuring the underlying loss landscape geometry. The vector n is frozen throughout and b is either frozen at its post-bias-correction value or left free depending on the experiment.

The perturbation shifts the effective parameters by exactly $(\Delta R_{\text{eff}}, \Delta M_{\text{eff}})$:

$$m \leftarrow m + \Delta R_{\text{eff}} \cdot \hat{n}, \quad W_{\text{in}} \leftarrow W_{\text{in}} + \Delta M_{\text{eff}} \cdot \hat{n}. \quad (21)$$

We test $R_{\text{eff}} \in \{1.3, 1.5, 1.7\}$, each $\varepsilon = 0.3$ above and below $M_{\text{crit}}(R_{\text{eff}})$, spanning from close to the bifurcation ($R_{\text{eff}} = 1.3$) to strongly bistable ($R_{\text{eff}} = 1.7$).

Three retraining strategies are compared across the same six initialisations. Experiment 1 skips the bias correction and leaves b free, so b_{eff} drifts throughout retraining. Experiment 2 applies the bias correction, freezes b , and uses a single RMSprop learning rate of 2×10^{-3} for both m and W_{in} ; this is the corrected baseline. Experiment 3 matches Experiment 2 but uses separate learning rates: 2×10^{-3} for m and 10^{-2} for W_{in} , giving M_{eff} five times faster ascent than R_{eff} . Results are in Section 5.4.

5 Results

This section presents four sets of experimental results. We first compare base training with b free and $b = 0$ frozen, providing the empirical foundation for the argument of Section 4.4. We then test whether a good initialisation alone can rescue $b = 0$ training, showing that it cannot when n rotates freely. We then present the main retraining experiments, in which networks initialised on either side of the flip boundary $M_{\text{crit}}(R_{\text{eff}})$ are retrained under three different strategies, and we compare their early-step dynamics in detail.

5.1 Base training: bias free versus bias frozen

Figure 2 shows the two base-training trajectories in the $(R_{\text{eff}}, M_{\text{eff}})$ plane, overlaid on the gradient field and the loss landscape. With b free (blue), the trajectory enters the bistable region $R_{\text{eff}} > 1$ within the first 1000 epochs and converges to $(R_{\text{eff}}, M_{\text{eff}}) \approx (1.50, 7.60)$, well inside the strongly bistable regime. With $b = 0$ frozen (red), the trajectory reaches $R_{\text{eff}} = 1.5$ before drifting back toward a monostable high-gain solution at $(R_{\text{eff}}, M_{\text{eff}}) \approx (0.85, 6.2)$. Both runs achieve comparably low training loss in the loss-curve panel, but the loss alone disguises the qualitative difference as only the bias-free network has crossed the bifurcation and acquired two memory states. The red-trajectory final position lies on the monostable side of the $R_{\text{eff}} = 1$ boundary, where the network performs the task by driving state transitions via large input gain rather than by storing the input in an attractor.

To rule out the possibility that the $b = 0$ failure is an artefact of a particular learning-rate or seed choice, we ran a sweep of 5 seeds $\times$ 4 learning rates (20 runs total). The results are summarised in Table 1. Only the smallest learning rate reaches $R_{\text{eff}} > 1$ at all, and even there the network barely crosses the bifurcation ($R_{\text{eff}} \in [1.01, 1.10]$ across five seeds, well short of the $R_{\text{eff}} \approx 1.50$ achieved with b free). All other settings find the monostable high-gain solution or diverge, confirming that the symmetry-breaking effect

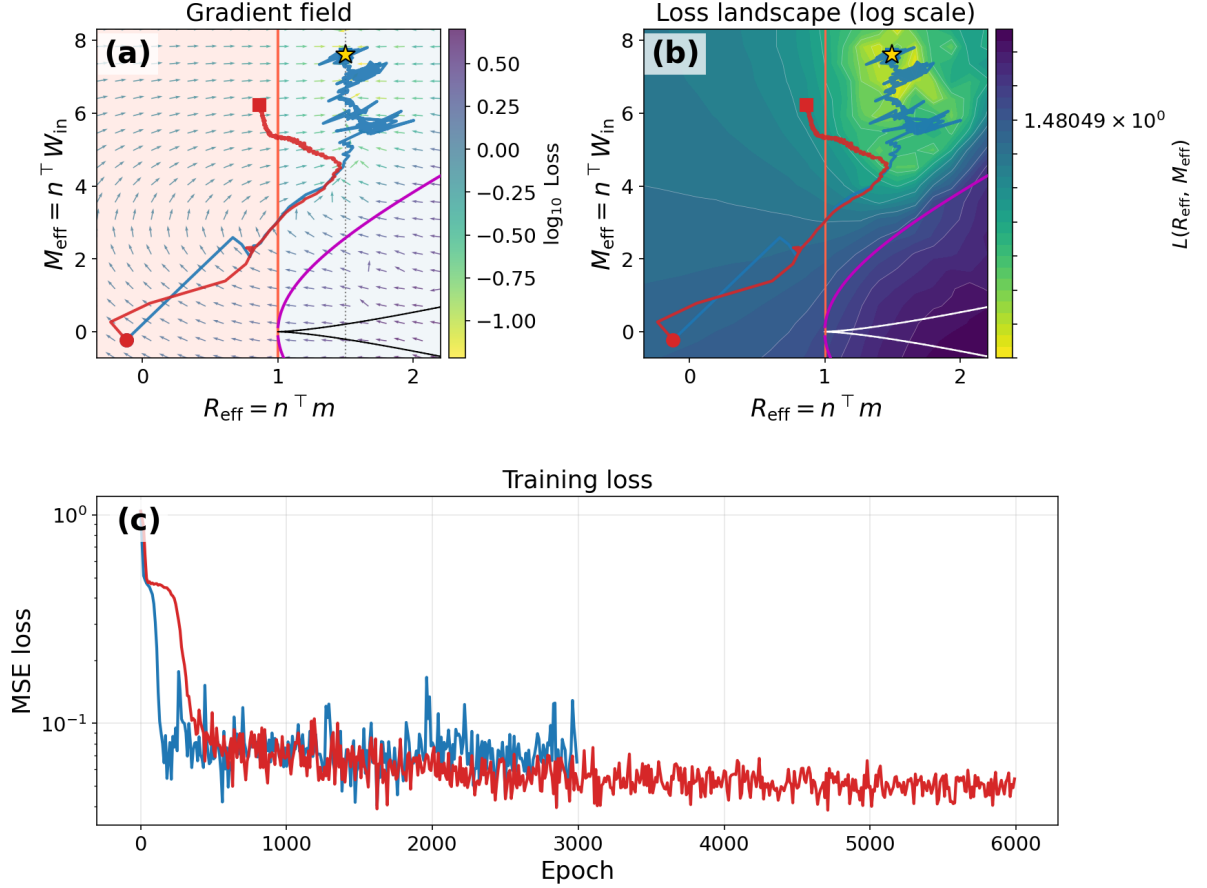


Figure 2: Base training with b free (blue) versus $b = 0$ frozen (red). Circles mark the start and squares the end of each trajectory. (a) Gradient field in $(R_{\text{eff}}, M_{\text{eff}})$ space: the magenta curve is the flip boundary $M_{\text{crit}}(R_{\text{eff}})$; the shaded red region is the monostable regime ($R_{\text{eff}} < 1$). With b free the trajectory enters the bistable region and converges to $(R_{\text{eff}}, M_{\text{eff}}) \approx (1.50, 7.60)$; with $b = 0$ frozen it reaches $R_{\text{eff}} = 1.5$ before drifting to a monostable solution. (b) Log-scale loss landscape with the same trajectories overlaid. (c) Training loss versus epoch for both conditions. Both endpoints achieve comparable MSE (≈ 0.05), illustrating that training loss alone does not distinguish bistable from monostable solutions.

of the bias is essential for reliable convergence to a strongly bistable solution.

lr	seed	R_{eff}	M_{eff}	loss	bistable
1e-3	0	1.022	5.558	0.049	✓
1e-3	1	1.099	4.670	0.053	✓
1e-3	2	1.051	5.503	0.057	✓
1e-3	3	1.034	5.253	0.053	✓
1e-3	4	1.011	4.342	0.051	✓
3e-3	0	0.864	6.227	0.050	×
... (all 5 seeds: $R_{\text{eff}} \in [0.66, 0.88]$, monostable)					
lr = 1e-2: monostable, $R_{\text{eff}} < 1$ (often near 0 or negative)					
lr = 3e-2: diverged or degenerate (NaN or $ M_{\text{eff}} \gg 10$)					

Table 1: Base training with $b = 0$ frozen, 5 seeds $\times$ 4 learning rates (6 000 epochs each, Adam). Only lr=1e-3 reaches $R_{\text{eff}} > 1$, but barely ($R_{\text{eff}} \in [1.01, 1.10]$ versus the target $R_{\text{eff}} \approx 1.50$ reached with b free); all other settings find the monostable high-gain solution or diverge.

5.2 Initialisation experiment: is a good start enough?

We next address the natural follow-up question: if the failure of $b = 0$ training is a problem of finding the bistable region, would starting inside it suffice? Figure 3 shows base training from four initialisations with $b = 0$ frozen: zero ($R_{\text{eff}} = 0$, $M_{\text{eff}} = 0$), small ($R_{\text{eff}} = 0.1$, $M_{\text{eff}} = 0.1$), normal ($R_{\text{eff}} = 1.0$, $M_{\text{eff}} = 1.0$), and bistable ($R_{\text{eff}} = 1.1$, $M_{\text{eff}} = 1.41$). None of the four converges to the base model’s bistable attractor at $(R_{\text{eff}}, M_{\text{eff}}) \approx (1.5, 7.6)$. The three initialisations starting in or near the monostable region go to the bistable region first but then drift back toward the monostable region, and most surprisingly, the initialization started inside the bistable region cannot choose a direction toward monostable or bistable attractor and just circles in place.

Two factors are responsible. First, the M_{eff} symmetry of the loss under $M_{\text{eff}} \rightarrow -M_{\text{eff}}$, with bias frozen at zero, allows gradient descent to drift in M_{eff} but does not provide a pull toward any particular bistable target. Second, and more importantly, the readout vector n is itself a free parameter during base training. As n rotates, the coordinates

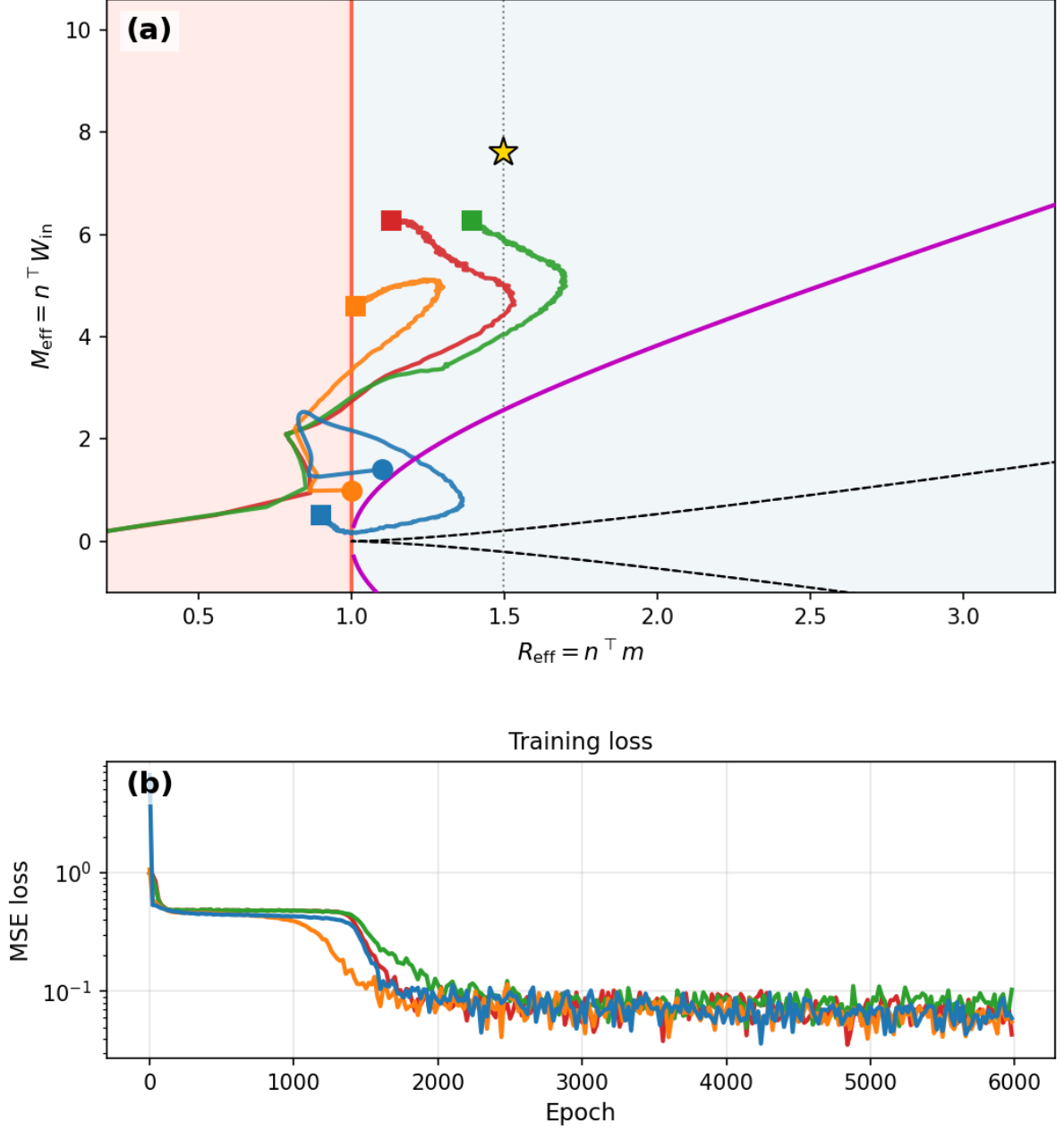


Figure 3: Base training from four initialisations with $b = 0$ frozen: **(a)** zero ($R_{\text{eff}} = 0$, $M_{\text{eff}} = 0$), small ($R_{\text{eff}} = 0.1$, $M_{\text{eff}} = 0.1$), normal ($R_{\text{eff}} = 1.0$, $M_{\text{eff}} = 1.0$), and bistable ($R_{\text{eff}} = 1.1$, above $M_{\text{crit}}(1.1)$). Even the bistable initialization cannot stay in bistable region because n rotates freely during base training and no gradient projection is applied, the optimiser finds a non-bistable solution along a different axis. The phase-portrait trajectories here are measured in a *moving* coordinate frame and are therefore not directly comparable to the fixed-frame retraining trajectories of Experiments 1–3 in the following subsection. **(b)** All runs achieve low MSE loss.

$R_{\text{eff}} = n^\top m$ and $M_{\text{eff}} = n^\top W_{\text{in}}$ are measured along a moving axis; the trajectory we see plotted in the figure is the projection of the actual N -dimensional optimisation onto whatever n is at each epoch. The $N - 1$ components of m and W_{in} orthogonal to n are unconstrained and, in the absence of any term in the loss that penalises movement away from the bistable axis, the optimiser freely moves into them, producing a solution that performs the task by an essentially different mechanism. The final $R_{\text{eff}} < 1$ visible in the figure is the projection of that different mechanism onto the final (rotated) n , not a record of a trajectory that stayed within the original phase plane.

This result complements the symmetry argument of Section 4.4, bias is needed not merely because the loss is symmetric under $M_{\text{eff}} \rightarrow -M_{\text{eff}}$, but because in its absence the optimiser has no force keeping it on the rank-1 axis defined by the initial n . Initialisation cannot substitute for bias because the optimiser is not constrained to respect the coordinate system in which the initialisation was chosen.

5.3 Strong-synapse initialisation removes the need for bias

The symmetry argument of Section 4.4 predicts that bias is needed only because standard initialisation places the network at the symmetric point $M_{\text{eff}} = 0$, where $\partial L / \partial M_{\text{eff}} = 0$. This suggests that any initialisation starting the network away from that point should break the symmetry just as bias does, even with $b = 0$ frozen.

We test this by rescaling the readout vector. With standard initialisation $m, n \sim \mathcal{N}(0, 1/N)$, the effective parameters satisfy $\text{Var}(R_{\text{eff}}) = N \cdot \frac{1}{N} \cdot \frac{1}{N} = 1/N$, so $R_{\text{eff}} \approx 0$ at initialisation. If instead $n \sim \mathcal{N}(0, 1)$ while keeping $m \sim \mathcal{N}(0, 1/N)$, then $\text{Var}(R_{\text{eff}}) = N \cdot 1 \cdot \frac{1}{N} = 1$ and $\text{Var}(M_{\text{eff}}) = N \cdot 1 \cdot \frac{1}{N} = 1$, so both effective parameters start at $O(1)$ and the network begins away from the symmetric point.

Table 2 compares the two initialisations under identical $b = 0$ frozen training (Adam, $\text{lr} = 10^{-3}$, 6 000 epochs, 5 seeds). Standard initialisation reaches only $R_{\text{eff}} \in [1.02, 1.12]$, confirming the barely-bistable outcome of Section 5.2. Strong-synapse initialisation reaches

$R_{\text{eff}} \in [1.24, 1.72]$ (mean ≈ 1.49) across all five seeds, attaining the target $R_{\text{eff}} \approx 1.5$ that otherwise required the bias term. The strongly-initialised networks converge to a lower input gain ($M_{\text{eff}} \approx 2.1$ versus ≈ 5.3 for standard init), but remain bistable in every run.

Init	Seed	R_{init}	M_{init}	R_{fin}	M_{fin}	Bistable
standard $n \sim \mathcal{N}(0, 1/N)$	0	0.257	0.054	1.095	5.245	✓
standard $n \sim \mathcal{N}(0, 1/N)$	1	-0.051	-0.043	1.111	5.887	✓
standard $n \sim \mathcal{N}(0, 1/N)$	2	0.059	-0.053	1.117	6.077	✓
standard $n \sim \mathcal{N}(0, 1/N)$	3	-0.022	-0.109	1.061	4.862	✓
standard $n \sim \mathcal{N}(0, 1/N)$	4	0.068	-0.029	1.021	4.504	✓
strong $n \sim \mathcal{N}(0, 1)$	0	2.573	0.537	1.244	3.864	✓
strong $n \sim \mathcal{N}(0, 1)$	1	-0.509	-0.426	1.630	1.458	✓
strong $n \sim \mathcal{N}(0, 1)$	2	0.589	-0.532	1.566	1.608	✓
strong $n \sim \mathcal{N}(0, 1)$	3	-0.216	-1.093	1.294	1.259	✓
strong $n \sim \mathcal{N}(0, 1)$	4	0.680	-0.286	1.719	2.105	✓

Table 2: Base training with $b = 0$ frozen under standard versus strong-synapse initialisation (Adam, lr = 10^{-3} , 6 000 epochs). Standard init reaches only $R_{\text{eff}} \approx 1.1$; strong-synapse init reaches $R_{\text{eff}} \approx 1.5$ in every seed without any bias term.

This confirms that bias is not uniquely necessary, it is one of at least two ways to break the M_{eff} symmetry. Under standard small-weight initialisation the network sits at the symmetric point and requires bias to escape it; under strong-synapse initialisation the network starts away from that point and reaches the bistable target on its own. The role of bias is therefore specific to the small-weight regime, not a general requirement for bistable training.

5.4 Retraining experiments: above and below the flip boundary

We now turn to the central experiment of the thesis. Starting from the corrected base model ($b_{\text{eff}} = 0$, b frozen), we test whether retraining trajectories initialised above and below the flip boundary $M_{\text{crit}}(R_{\text{eff}})$ behave qualitatively differently, as the bifurcation analysis predicts. We test three values of $R_{\text{eff}} \in \{1.3, 1.5, 1.7\}$ spanning the bistable regime, each $\varepsilon = 0.3$ above and below $M_{\text{crit}}(R_{\text{eff}})$, and we retrain for 2 000 RMSprop steps

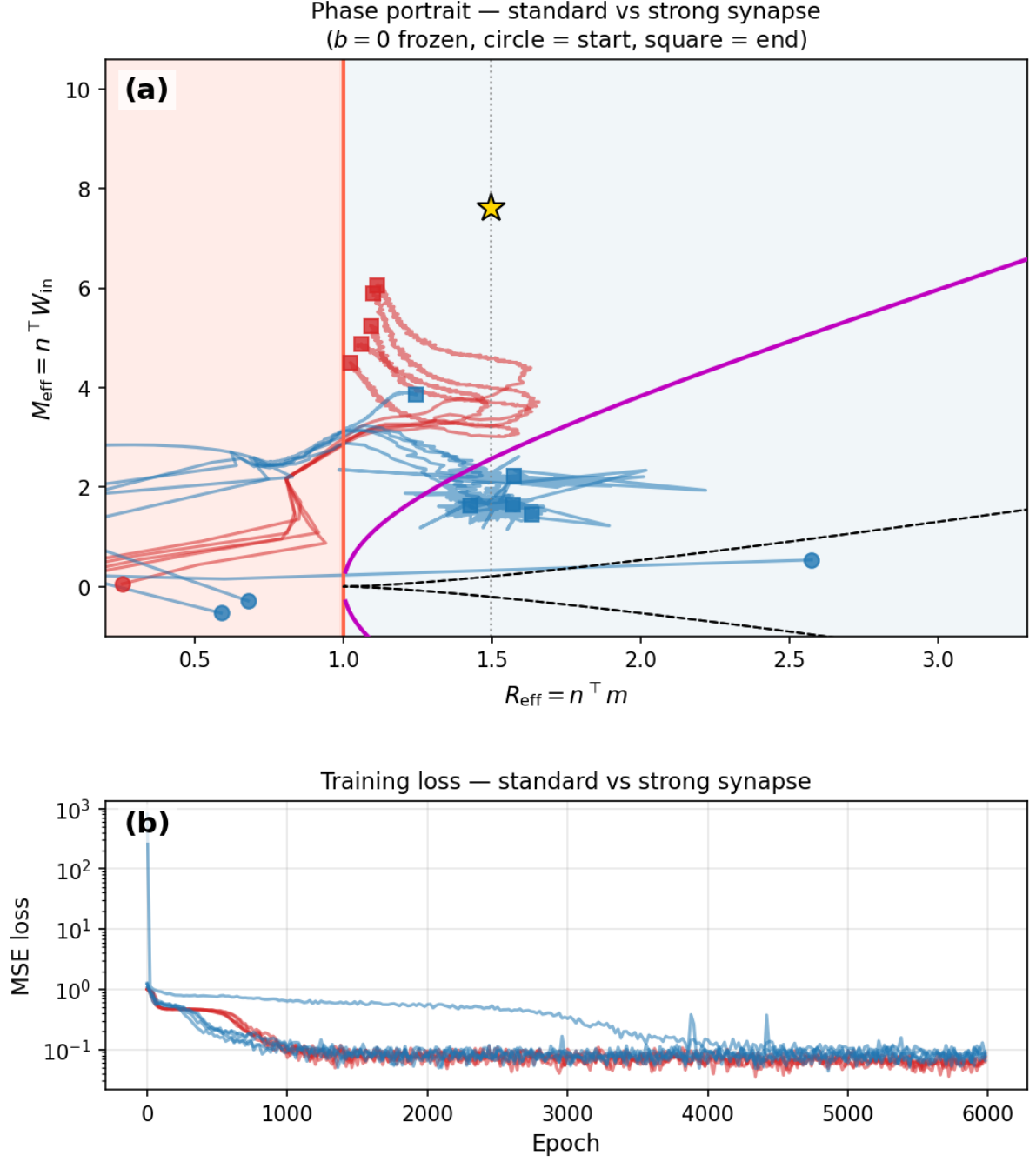


Figure 4: Standard ($n \sim \mathcal{N}(0, 1/N)$, red) versus strong-synapse ($n \sim \mathcal{N}(0, 1)$, blue) initialisation with $b = 0$ frozen. **(a)** phase portrait; circles mark initialisation, squares convergence. Strong init reaches higher R_{eff} at lower M_{eff} . **(b)** loss curves; strong init converges more slowly but reaches comparable loss.

under three different conditions (Experiments 1, 2, 3 of Section 4.6).

Experiment 1: bias not corrected. Figure 5 shows the same six initialisations under a naive setup in which the b_{eff} correction is not applied and b remains a free parameter during retraining. The most visible difference is in the trajectory shapes: convergence is markedly noisier than in Experiment 2, with visible zigzagging near the attractor. The cause is that as b updates during retraining, the residual $b_{\text{eff}} = n^\top b$ drifts, continuously shifting the effective flip boundary on which the network’s behaviour depends. The final positions in Experiment 1 land in a good stable region but the path toward the end is messy.

Experiment 2: corrected baseline. Figure 6 shows the corrected baseline (bias frozen at $b_{\text{eff}} = 0$) results. Three observations are consistent with the bifurcation prediction. Initialisations above the flip boundary move smoothly through the bistable region, climbing in M_{eff} while R_{eff} stays well above 1, and converge to a common region near $(R_{\text{eff}}, M_{\text{eff}}) \approx (1.5, 6.0)$, with M_{eff} ranging from about 5.85 at the lowest tested R_{eff} to about 6.2 at the highest. Below-boundary initialisations at $R_{\text{eff}} = 1.3$ (the dashed orange trajectory) momentarily cross below $R_{\text{eff}} = 1$ and temporarily lose bistability before recovering—the trajectory dips down and to the left before turning around and climbing back into the bistable region. This is the optimisation analog of a network that has acquired a memory and then briefly forgotten it during an unstable phase of training. At $R_{\text{eff}} = 1.5$ and 1.7, both above and below trajectories converge to the same final region, but the below-boundary trajectories take a more curved path through lower M_{eff} values before climbing to the attractor. The deviation of the trajectories from the bifurcation prediction is largest at $R_{\text{eff}} = 1.3$, where the network sits closest to the pitchfork bifurcation; the model reduction, which assumes a well-developed bistable attractor, is least accurate there.

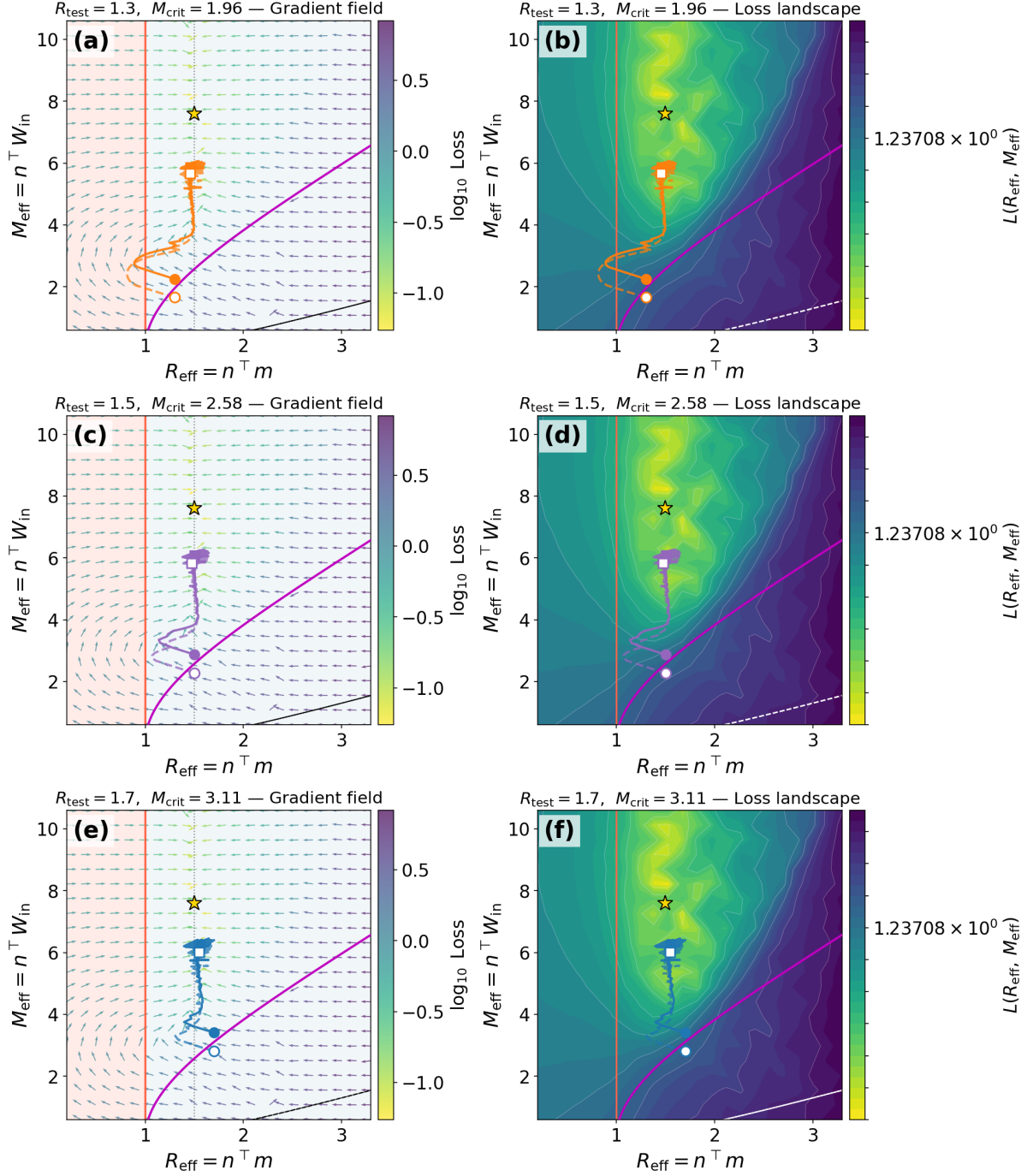


Figure 5: Experiment 1: b free during retraining, no b_{eff} correction. (a)(c)(e) gradient field with the flip boundary (magenta) and fold curves (dashed black). (b)(d)(f) log-scale loss landscape with the same overlays. Trajectories exhibit visible zigzagging near convergence as b_{eff} drifts throughout optimisation, continuously shifting the effective flip boundary.

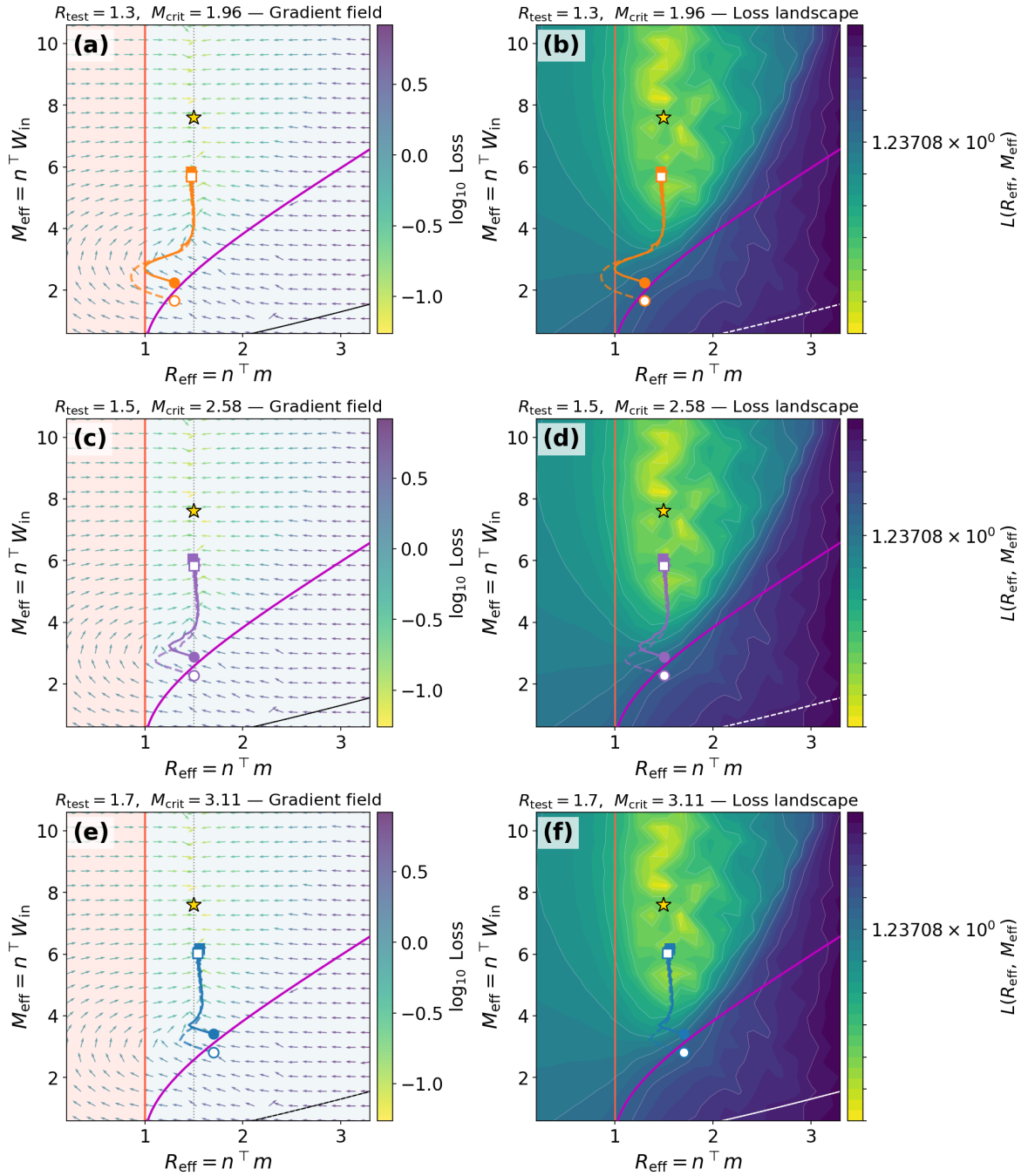


Figure 6: Experiment 2: corrected baseline with $b_{\text{eff}} = 0$ and b frozen. Above-boundary initialisations climb smoothly through the bistable region; below-boundary initialisations at $R_{\text{eff}} = 1.3$ transiently lose bistability before recovering.

Experiment 3: separate learning rates. Figure 7 shows results when m and W_{in} use separate RMSprop optimisers with different learning rates: $\eta_m = 2 \times 10^{-3}$ for m which controls R_{eff} and $\eta_{W_{\text{in}}} = 10^{-2}$ for W_{in} which controls M_{eff} . The five-times-faster M_{eff} ascent visibly changes the early-step dynamics: M_{eff} climbs rapidly out of the boundary region in the first few steps, pulling R_{eff} along behind it, and the resulting trajectories curve less toward the $R_{\text{eff}} = 1$ boundary than in Experiment 2.

All three experiments converge to the same narrow region at $R_{\text{eff}} \approx 1.45\text{--}1.55$ and $M_{\text{eff}} \approx 5.8\text{--}6.2$, regardless of initialisation side, retraining strategy, or learning rate — significantly below the base model’s $M_{\text{eff}} \approx 7.6$. This ceiling arises because gradient projection freezes the $N - 1$ components of m and W_{in} orthogonal to n , shifting the constrained minimum of the projected loss away from the unconstrained base-training optimum. The fact that all three strategies reach the same destination confirms the ceiling is a property of the loss landscape rather than the optimiser.

Although Experiments 2 and 3 converge to the same location, their early-step dynamics differ substantially, as shown in Table 3 for the $R_{\text{eff}} = 1.3$ above-boundary trajectory. Both start with the same drop in R_{eff} from 1.300 to 1.081 at step 1, but M_{eff} jumps to 3.35 in Experiment 3 versus 2.47 in Experiment 2. By step 5, Experiment 3 has reached $R_{\text{eff}} = 1.387$ with loss 0.19 while Experiment 2 is still at $R_{\text{eff}} = 1.006$ with loss 0.93. The faster W_{in} learning rate pushes M_{eff} above the flip boundary in the first few steps, pulling R_{eff} back into the bistable region before it can drift toward $R_{\text{eff}} = 1$. By step 500 both experiments are indistinguishable.

5.5 Short summary of empirical findings

The five sets of experiments together support four claims from the analysis of Section 4. First, base training without bias fails to reach the bistable regime under standard small-weight initialisation, and this failure cannot be rescued by repositioning inside the bistable region at fixed weight scale; strong-synapse initialisation removes the need for

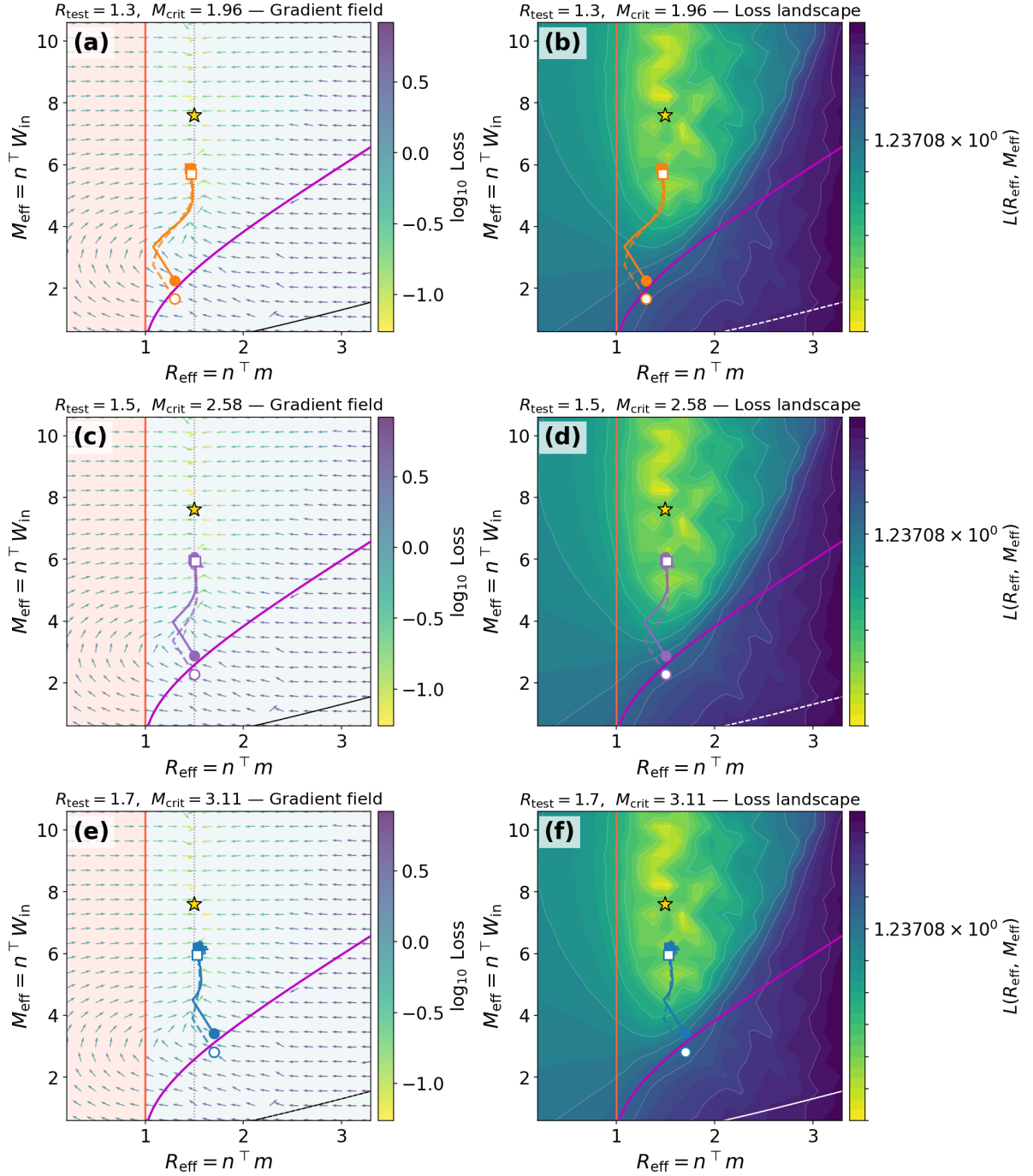


Figure 7: Experiment 3: separate learning rates for m and W_{in} . Trajectories reach target R_{eff} values significantly faster in the first 30 steps compared to Experiment 2 (see Table 3), and curve less toward the bifurcation $R_{\text{eff}} = 1$, yielding more direct paths in the $(R_{\text{eff}}, M_{\text{eff}})$ plane. The final asymptotic location is essentially the same as in Experiment 2, indicating that the ceiling at $M_{\text{eff}} \approx 6$ is a property of the projected loss landscape rather than an optimiser artefact.

Step	Experiment 2 (single lr=2e-3)			Experiment 3 (W_{in} lr=1e-2)		
	R_{eff}	M_{eff}	loss	R_{eff}	M_{eff}	loss
0	1.300	2.26	1.572	1.300	2.26	1.570
1	1.081	2.47	1.100	1.081	3.35	0.657
5	1.006	2.78	0.926	1.387	4.27	0.188
10	1.226	3.11	0.655	1.467	4.73	0.148
20	1.449	3.64	0.322	1.480	5.20	0.129
29	1.487	3.88	0.236	1.474	5.43	0.112
500	1.483	5.84	0.106	1.441	5.84	0.093
1999	1.483	5.85	0.104	1.493	5.91	0.105

Table 3: Step-by-step comparison for the $R_{\text{eff}} = 1.3$, above-boundary trajectory. At step 1 both experiments share the same R_{eff} drop ($1.300 \rightarrow 1.081$), but M_{eff} jumps to 3.35 in Experiment 3 versus 2.47 in Experiment 2 because the learning rate on W_{in} is five times larger. By step 5, Experiment 3 has reached $R_{\text{eff}} = 1.387$ with loss 0.19 while Experiment 2 is still at $R_{\text{eff}} = 1.006$ with loss 0.93. Both converge to the same ceiling by step 500.

bias, confirming the failure is specific to the small-weight regime. Second, retraining trajectories above and below the flip boundary $M_{\text{crit}}(R_{\text{eff}})$ behave differently, with the clearest distinction at $R_{\text{eff}} = 1.3$ where the below-boundary trajectory transiently loses bistability before recovering. Third, all three retraining strategies converge to the same $(R_{\text{eff}}, M_{\text{eff}})$ neighbourhood, confirming that the convergence ceiling is a property of the projected loss landscape rather than the optimiser. Fourth, separate learning rates for m and W_{in} improve early-step dynamics without changing the asymptotic destination.

6 Discussion

The experiments validate the model reduction part of Section 4 and raise several points worth addressing beyond what the results show directly.

6.1 Bias as a symmetry-breaker

The bias vector has no role in the fixed-point structure as bistability holds for any $R_{\text{eff}} > 1$ regardless of b , yet its absence during base training under standard small-weight initialisation consistently prevents the network from reaching the bistable regime. The explanation is a symmetry argument: without bias the loss is even in M_{eff} , so gradient descent has no signal to move in the direction it needs to and settles for a monostable high-gain solution instead. A small bias breaks this symmetry from the first epoch. This kind of symmetry-breaking via a small structural term has similarities with the broader theory of learning dynamics [Saxe et al., 2014].

A practical implication is that bias is only needed during base training, and even then only in the small-weight regime. Section 5.3 shows that strong-synapse initialisation reaches the bistable target with $b = 0$ frozen, confirming that bias and initialisation scale are two routes to the same end: breaking the M_{eff} symmetry so that gradient descent receives a signal from the first epoch.

6.2 The moving coordinate-frame problem

The initialisation experiment shows that placing the network inside the bistable region does not guarantee gradient descent stays there, because $(R_{\text{eff}}, M_{\text{eff}})$ depends on n and n rotates freely during base training. The trajectory one observes in the $(R_{\text{eff}}, M_{\text{eff}})$ plane is therefore the projection of an N -dimensional path onto a moving axis and is not a faithful record of the network’s position in the analysed coordinate system.

This has a broader methodological implication. Any low-dimensional coordinate used

to track training: PCA of weights, fixed-point snapshots, or the $(R_{\text{eff}}, M_{\text{eff}})$ projection used here is only meaningful if the basis defining that coordinate is held fixed. Freezing n and projecting gradients onto $\hat{n}$ is the simplest way to enforce this in the rank-1 setting; in higher-rank settings the natural generalisation is to freeze the rank-decomposition basis or equivalently the low-dimensional subspace into which the effective parameters project.

6.3 What the rank-1 analysis is good for

Rank-1 RNNs do not solve interesting tasks. Their value is that they are the smallest member of the low-rank family, small enough that every training question can be answered by hand. When the field disagrees about whether bias matters, whether a good initialisation is sufficient, or whether a learning-rate heuristic changes asymptotic behaviour or only transients, the rank-1 case is where the answer can be derived rather than measured. The present work provides such a derivation in a form complete enough to serve as a benchmark for the same questions in larger networks.

6.4 Limitations and future directions

Three limitations follow from the narrow scope of this work, each suggesting a concrete next step.

The first is that the one-bit flip-flop requires only rank-1. The three-bit version of Sussillo and Barak [2013] requires rank-3, and the model reduction generalises in principle to a rank- R collective coordinate κ governed by an effective parameter matrix $\mathbf{R}_{\text{eff}} \in \mathbb{R}^{R \times R}$ with entries $n^{(r)\top} m^{(s)}$, with a corresponding multi-dimensional flip boundary. The rank-1 analysis is the building block needed to start this generalisation.

The second is that bistability is one motif among many. Driscoll et al. [2024] catalogue ring attractors, line attractors, decision boundaries, and rotational structures, each with its own bifurcation structure and natural minimal-rank model. A natural research programme would derive the flip-boundary analog for each motif and test whether it predicts training

success in the corresponding full-rank network.

The third, and most important, is the extension to full-rank networks. Schuessler et al. [2020] show that training on low-dimensional tasks induces approximately low-rank changes to the connectivity, suggesting the present framework should apply to the leading rank-1 direction of a full-rank trained network. The convergence ceiling at $M_{\text{eff}} \approx 6$ being well below the unconstrained base-model value of 7.6 is already a hint that the input gain is partly absorbed by connectivity directions that the rank-1 analysis does not track, consistent with this picture. The natural next experiment is to repeat the bias and initialisation analysis on a network with $W_{\text{rec}} = W_0 + mn^\top$, where W_0 is a fixed random component and (m, n) are the trainable low-rank correction. If the symmetry-breaking role of bias and the necessity of n -freezing survive the addition of W_0 , the framework extends; if they do not, the discrepancies would themselves be informative.

A further limitation is methodological. The retraining trajectories in Section 5 are produced under gradient projection onto the readout direction $\hat{n}$, which confines optimisation to the two effective degrees of freedom $(R_{\text{eff}}, M_{\text{eff}})$. This makes the trajectories directly comparable to the reduced model, but it also means the network is effectively updating only two parameters rather than exploring the full N -dimensional landscape. The clean two-dimensional dynamics we observe are therefore partly a consequence of the constraint, not solely a property of the loss surface. A natural next step is to track $(R_{\text{eff}}, M_{\text{eff}})$ as observables under unconstrained gradient descent, where n is free to rotate and the orthogonal directions of m and W_{in} are updated; comparing the resulting trajectories to the projected ones would reveal how much of the picture survives outside the rank-1 constraint.

7 Conclusion

This thesis has presented an analysis of the rank-1 RNN trained on a one-bit flip-flop task, together with experiments that test the predictions of that analysis directly.

The central analytical result is a two-parameter reduction $(R_{\text{eff}}, M_{\text{eff}})$, with $R_{\text{eff}} = n^\top m$ controlling the pitchfork bifurcation that produces bistability and $M_{\text{eff}} = n^\top W_{\text{in}}$ controlling whether an input pulse is strong enough to switch the network’s memory state in a single step. The flip boundary $M_{\text{crit}}(R_{\text{eff}})$ separating flip-capable from flip-incapable regions can be derived in closed form, and the entire training dynamics can be plotted as a trajectory in this two-dimensional plane on top of an analytically predicted loss landscape.

Three empirical findings follow. First, bias acts as a symmetry-breaker during base training under standard small-weight initialisation. Without it, gradient descent fails to reach the bistable regime under all hyperparameter combinations tested, and repositioning inside the bistable region at fixed weight scale does not help because n rotates freely off the analysed axis. This failure is specific to the small-weight regime, strong-synapse initialisation breaks the same symmetry and reaches the bistable target without any bias term. Second, retraining trajectories above and below the flip boundary behave qualitatively differently, most clearly at $R_{\text{eff}} = 1.3$ among the ones we tested where the below-boundary trajectory transiently loses bistability before recovering. Third, the convergence ceiling at $M_{\text{eff}} \approx 6$ is a property of the projected loss landscape rather than the optimiser; separate learning rates change the early-step dynamics but not the asymptotic destination, confirming that giving W_{in} a faster learning rate than m is a useful heuristic for staying in the bistable region during early retraining without affecting where the network ultimately converges.

We have studied a single dynamical motif (bistability) at the smallest rank in the low-rank family that supports it, on the smallest task that requires it. The value of this narrow scope is not that rank-1 RNNs are interesting models in their own right, but that every question about their training can be answered by hand, isolating the conceptual

issues: bias as symmetry-breaker, the coordinate-frame subtlety, and the constrained versus unconstrained loss minimum from the dimensionality issues that would arise in a higher-rank treatment. The natural extensions are discussed in Section 6.4. With the rank-1 picture established, the same questions can be asked in larger settings against a known analytical baseline.

References

- Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Current Opinion in Neurobiology*, 46:1–6, 2017. doi: 10.1016/j.conb.2017.06.003.
- Manuel Beiran, Alexis Dubreuil, Adrian Valente, Francesca Mastrogiuseppe, and Srdjan Ostojic. Shaping dynamics with multiple populations in low-rank recurrent networks. *Neural Computation*, 33(6):1572–1615, 2021. doi: 10.1162/neco_a_01381.
- Fatih Dinc, Ege Cirakman, Yiqi Jiang, Mert Yuksekgonul, Mark J. Schnitzer, and Hidenori Tanaka. A ghost mechanism: An analytical model of abrupt learning in recurrent networks. *arXiv preprint arXiv:2501.02378*, 2025.
- Laura N. Driscoll, Krishna V. Shenoy, and David Sussillo. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *Nature Neuroscience*, 27: 1349–1363, 2024. doi: 10.1038/s41593-024-01668-6.
- Alexis Dubreuil, Adrian Valente, Manuel Beiran, Francesca Mastrogiuseppe, and Srdjan Ostojic. The role of population structure in computations through neural dynamics. *Nature Neuroscience*, 25(6):783–794, 2022. doi: 10.1038/s41593-022-01088-4.
- Scott Gigante, Adam S. Charles, Smita Krishnaswamy, and Gal Mishne. Visualizing the PHATE of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.
- Udith Haputhanthri, Liam Storan, Yiqi Jiang, Tarun Raheja, Adam Shai, Orhun Akengin, Nina Miolane, Mark J. Schnitzer, Fatih Dinc, and Hidenori Tanaka. Understanding and controlling the geometry of memory organization in RNNs. *arXiv preprint arXiv:2502.07256*, 2025.
- Renate Krause, Matthew Cook, Sepp Kollmorgen, Valerio Mante, and Giacomo Indiveri. Operative dimensions in unconstrained connectivity of recurrent neural networks. In

Advances in Neural Information Processing Systems (NeurIPS), volume 35, pages 17073–17085, 2022.

Gaspard Lambrechts, Florent De Geeter, Nicolas Vecoven, Damien Ernst, and Guillaume Drion. Warming up recurrent neural networks to maximise reachable multistability greatly improves learning. *Neural Networks*, 166:645–669, 2023. doi: 10.1016/j.neunet.2023.07.023.

Niru Maheswaranathan, Alex H. Williams, Matthew D. Golub, Surya Ganguli, and David Sussillo. Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019a.

Niru Maheswaranathan, Alex H. Williams, Matthew D. Golub, Surya Ganguli, and David Sussillo. Universality and individuality in neural dynamics across large populations of recurrent networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019b.

Valerio Mante, David Sussillo, Krishna V. Shenoy, and William T. Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474): 78–84, 2013. doi: 10.1038/nature12742.

Francesca Mastrogioiuseppe and Srdjan Ostojic. Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*, 99(3):609–623, 2018. doi: 10.1016/j.neuron.2018.07.003.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, volume 28, pages 1310–1318, 2013.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the non-

- linear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.
- Friedrich Schuessler, Francesca Mastrogiuseppe, Alexis Dubreuil, Srdjan Ostojic, and Omri Barak. The interplay between randomness and structure during learning in RNNs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.
- David Sussillo and Omri Barak. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3):626–649, 2013. doi: 10.1162/NECO_a_00409.
- Christopher Versteeg, Jonathan D. McCart, Mitchell Ostrow, David M. Zoltowski, Clayton B. Washington, Laura Driscoll, Olivier Codol, Jonathan A. Michaels, Scott W. Linderman, David Sussillo, and Chethan Pandarinath. Computation-through-dynamics benchmark: Simulated datasets and quality metrics for dynamical models of neural activity. *bioRxiv preprint 2025.02.07.637062*, 2025. doi: 10.1101/2025.02.07.637062.
- Saurabh Vyas, Matthew D. Golub, David Sussillo, and Krishna V. Shenoy. Computation through neural population dynamics. *Annual Review of Neuroscience*, 43:249–275, 2020. doi: 10.1146/annurev-neuro-092619-094115.
- Jiancheng Xie. Visualizing the PHATE of recurrent neural networks. Master’s thesis, Johns Hopkins University, 2024.
- Guangyu Robert Yang and Xiao-Jing Wang. Artificial neural networks for neuroscientists: A primer. *Neuron*, 107(6):1048–1070, 2020. doi: 10.1016/j.neuron.2020.09.005.
- Guangyu Robert Yang, Madhura R. Joglekar, H. Francis Song, William T. Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nature Neuroscience*, 22(2):297–306, 2019. doi: 10.1038/s41593-018-0310-2.