

Regularized vs. Unregularized RNN — Testing Biological Constraints for Robustness

Ethan Ganz

1. Introduction

Current AI innovation leans too heavily on scale and compute. The brain's system is far more cost- and energy-efficient than today's computers. If we could harness the brain's approach to processing and apply it in place of today's data centers, we could imagine a large, environmentally friendly computing center — unlike today's data centers — that would be far cheaper to run and would not drain nearly as much water or power.

Regularizing an RNN, or Recurrent Neural Network, does not change its power usage — it's meant to test whether it changes how robust the network is to noise. Specifically, we hypothesized that adding weight-cost and activity-cost penalties during training, inspired by the brain's own resource constraints, would make a small RNN more robust to noise.

We used the 3-bit flip-flop task as a testbed because of its small scale, as opposed to waiting weeks for results while using far more power and money on a larger-scale project. It is also much easier to find and analyze a small network's fixed points than it would be in a larger-scale project.

Our first finding was that both networks produced the same kind of internal structure: densely packed, non-isolated fixed points rather than 8 cleanly separated ones, regardless of regularization. Our second finding was that the difference between Network A and Network B was the size of the hidden-state activity that ran during processing, caused by Network B's activity-cost penalty. Correcting for this scale difference reversed the result: once corrected, Network B was consistently more robust to noise than Network A.

In this experiment, we trained two RNNs, characterized their fixed-point structure, and compared their robustness to noise before and after correcting for this hidden-state scale difference.

2. Methods

2.1 Task

For the 3-bit flip-flop task, we used 3 input channels, with random ± 1 pulses at a probability of 0.05. The output channels held the last sign until the next pulse flipped it. This task follows Sussillo & Barak (2013). We used a sequence length of $T=110$.

2.2 Networks

Our architecture used an RNN with a hidden size of 64, a linear output layer, and an input/output size of 3. Network A's training used 10,000 iterations, no regularization, MSE loss, and Adam. Network B's training used the same architecture, plus weight-cost and activity-cost penalty terms in the loss — λ_{weight} and $\lambda_{\text{activity}}$ were 0.01 and 0.001 respectively — for 25,000 iterations.

These two penalty terms were chosen because they correspond to real metabolic constraints on biological neurons: the weight-cost term reflects the wiring cost of a synapse — larger synaptic weights require more material and energy to build and maintain — while the activity-cost term reflects the spiking cost of a neuron, since each action potential consumes ATP to restore the membrane potential afterward. Penalizing both during training pushes the network toward solutions that are cheap along the same axes a biological network is actually constrained by, rather than toward an arbitrary notion of simplicity.

2.3 Fixed-point finder

A fixed point is a hidden state where, if you run the network one more step, the new hidden state is almost identical to the original. We started from 2,200 random hidden states and used gradient descent to adjust each one so its speed became as small as possible, stopping once speed reached a rather small value, which was between $1e-7$ and $1e-8$. Once we had this data, we measured the nearest-neighbor distance between the resulting hidden states, checking whether points landed close together or spread apart. We used PCA to project the high-dimensional hidden states down to 2D so they could be plotted and viewed.

2.4 Noisy rollout and scale correction

Instead of running the whole sequence through the RNN in a single pass, we looped over each timestep, and at every step added new random Gaussian noise to the hidden state before feeding it into the next step. When both networks were tested with the same absolute noise, Network A and Network B's hidden states ran at different natural scales — about 0.19 vs. about 0.12 in magnitude, respectively. So the identical noise hit the smaller-scale network harder. We measured each network's own average hidden-state magnitude, then scaled the noise for each network individually, using $\text{scaled_noise} = \text{magnitude} \times \text{noise_std}$. So both networks were tested with noise proportional to their own typical activity, not the same fixed amount.

3. Results

We searched from 2,200 random hidden states and found a final speed of $1e-7$ to $1e-8$. Over 92% of points landed within 0.01 of each other, and both networks showed the same pattern. As the noise levels (0, 0.05, 0.1, 0.15, 0.2) increased, both networks' loss went up, but Network B's loss rose faster than Network A's. At the highest noise level, Network A's loss was ~ 0.53 – 0.57 , while Network B's was ~ 0.73 – 0.77 — Network B was worse than Network A, the opposite of the hypothesis. Network A's average hidden-state magnitude was ~ 0.19 – 0.195 , Network B's was ~ 0.12 – 0.124 , and Network B's activity-cost penalty caused it to run at a smaller internal scale. After correcting for this scale difference, Network A's losses were [0.0088, 0.0102, 0.0128, 0.0205, 0.0330], while Network B's were [0.0023, 0.0048, 0.0085, 0.0129, 0.0198]. Network B had lower loss than Network A at every noise level, and

this pattern reproduced consistently across three separate runs with fresh random data each time.

4. Discussion

The dense fixed-point structure likely reflects the task's demand for sustained memory rather than regularization, because holding a memory state for many timesteps is easier with a wide region of nearby stable points (like a ball resting in a bowl) than with a single exact point (like a ball balanced on a needle) — and both networks, regardless of regularization, need to hold memory this way.

If a researcher recreated this experiment, after only the raw sweep they would choose Network A as the more robust network, while from the scaled sweep, the more robust choice would be Network B — a complete reversal of the conclusion depending on whether the scale correction is applied.

This study has several limitations. Because we only tested one task (the flip-flop memory task), these results might not hold for different memory tasks. Because the network was small (only 64 hidden units), it's possible a larger network with more capacity would show different fixed-point structure or robustness patterns, so these results may not hold at larger scales. Because we only trained one version of each network, we don't have a measure of how much these results vary by chance, so we can't be fully confident the pattern would hold if the networks were retrained from different random starting points. Despite the motivating framing about brain efficiency, we never directly measured real neurons, so this connection to the brain remains speculative.

5. Conclusion

We trained two RNNs that were identical except that one (Network B) included weight-cost and activity-cost regularization penalties during training, and compared both networks on their fixed-point structure and their robustness to noise. This is a small but personally significant proof of my main theory — that biologically-inspired constraints can make systems more efficient and robust without needing more scale — though given this study's limitations, it should be read as a small, suggestive step rather than a defining proof. The next step would be to study this further: in a few months, revisiting this project at a larger scale, on a better computer, and across multiple tasks.

References

Sussillo, D., & Barak, O. (2013). Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3), 626–649.