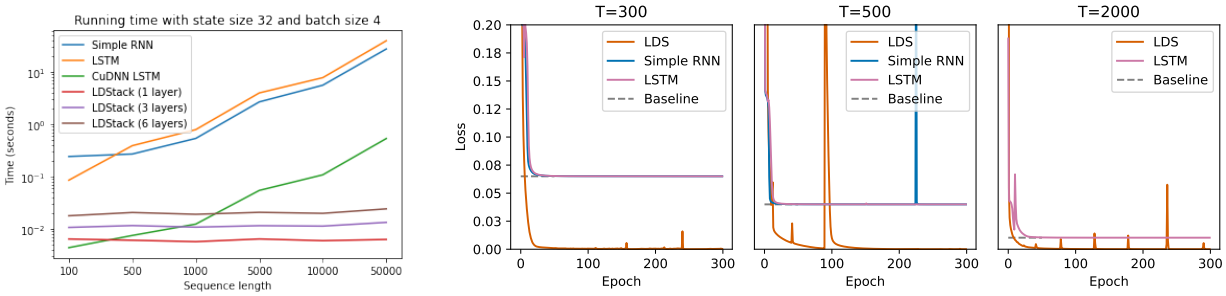

We thank all the reviewers for their careful attention to our paper, and their helpful comments.

Speed. See above Figure 1 (top left) for runtime comparison among RNNs. LDSTACK (research code in both Python and CUDA) is always faster than unoptimized RNNs. At longer sequence lengths, it is even faster than the highly-optimized, fused CuDNN LSTM. We didn't initially report this comparison because we will release an improved LDSTACK implementation, and expect it will be faster than CuDNN LSTM at essentially all problem sizes. Currently, LDSTACK begins with an expensive transpose, since our CUDA op takes time-major sequences. We use homemade (unoptimized) prefix scans and reductions, rather than those in the CUB library. Finally, the LDSTACK op can be fused.

Exposition in Sections 5 and 6. LDSTACK has a simple intuition, summarized as follows. The first layer is a plain LDS. Subsequently, at layer i, the previous layer's states $s^{(i-1)}$ are used to estimate where linear transitions incorrectly deviate from nonlinear ones. This is measured by $p_t^{(i-1)} = \delta(As_t^{(i-1)})$ where $\delta(a) = \rho(a)/a$ is the (multiplicative) deviation of the RNN's nonlinearity ρ. These correct the transitions within layer i via $s_{t+1}^{(i)} = \text{diag}(p_t^{(i-1)})As_t^{(i)} + Bx_t$. We regrettably removed this simpler explanation to fit the 8-page limit. We kept the current (rather abstract) exposition because we did not want to obscure the origins of the idea of stacking LDS. This was part of a concerted effort to correctly attribute ideas; insufficient discussion of related work was considered the major flaw of a previous submission. With the additional 9th page, we shall restore helpful explanations and full equations for LDSTACK. Similarly, we agree the experiments have condensed presentation, and deserve more details on the 9th page.

Discretization error. In order to prove correctness of LDSTACK, it is not necessary to pass to the original continuous-time scheme of Banks et al. In discrete time, the convergence of $s_t^{(i)}$ (uniformly across t) occurs at $i \leq T$. Once $s_t^{(i-1)}$ is correct (i.e. matches the nonlinear RNN), then $p_t^{(i-1)}$ perfectly corrects $s_{t+1}^{(i)}$. The first layer gets the first state correct, which makes the second layer get both the first and second states correct, and so on. (This is the "simple recursion" we briefly mentioned, and will of course elaborate with the available space.)

Long-term memory. Our MNIST experiment has short/medium-term dependencies. For long-term dependencies, it is useful to constrain A to have unit eigenvalues (as in an orthogonal or unitary matrix.) This constraint is trivial within our framework. Suppose the LDS eigenvalue λ has polar representation (r, θ). Then a real zero part of $\ln \lambda = \ln r + \theta i$ corresponds to eigenvalue magnitude $r = 1$. So, optimize over the θ of $\ln \lambda$ (with zero real part) rather than the real and imaginary parts of λ.

As shown in Figure 2 (top right), this solves the copying memory problem (Arjovsky et al., 2016). The goal is to remember the first 10 entries r of the input sequence, withhold output for T steps (for which the inputs are just "blanks"), and, upon seeing a "go" input at time $T + 10$, to output r. Unitary RNNs solve the problem, whereas standard RNNs do not outperform a trivial baseline. There is an LDS which achieves zero error (Henaff et al., 2016), so we don't consider multiple-layer LDSTACK. Arjovsky et al. use LSTM, simple tanh RNN, and uRNN of respective sizes $n = 40, 80$, and 128 for parameter counts of roughly 6500. We use $n = 160$ for the LDS, which has just 3380 parameters. The θ are initialized uniformly at random within $[-2\pi, 2\pi]$. We use the AdaMax optimizer with step size 0.01 and batch size 256 for 300 epochs on a sample size of 10,000. Our solution is the **state of the art**: it uses the simplest (linear) RNN with the fewest parameters to solve $T = 2000$, which demands full-capacity uRNNs (Wisdom et al., 2016) or later models.

Numerical stability. The LDS are indeed unstable at some λ. However, our initialization suggestion (λ as the roots of a monic polynomial with random coefficients) empirically avoids instability, even without techniques like gradient clipping. The calculation $\log B_i' = -\sum_{j \neq i} \log(1 - \lambda_j/\lambda_i)$ avoids high-degree powers and is (empirically) stable when all $\lambda_i \neq \lambda_j$ for $i \neq j$. This empirical success merits more formal/theoretical investigation.

Line 461 in Supplemental. Controllability indices are not affected by any *full rank* U transformation, and neither is normality of the induced distribution. Thank you for pointing this out, along with the other fixes.