Extended Abstract Track

Representational Drift as a Mechanism of Length-Generalization Failure

Andrew Mah AMAH@FLATIRONINSTITUTE.ORG
Flatiron Institute

Alex H. Williams AWILLIAMS@FLATIRONINSTITUTE.ORG
Flatiron Institute, New York University

Editors: List of editors' names

Abstract

Length generalization remains a challenge for recurrent sequence models. We identify a failure mode in linear RNNs in which task-relevant representations remain linearly decodable but drift coherently through hidden-state space over long sequences. Ablating the drift-associated dimension restores long-horizon performance without retraining, providing a geometric mechanism for unexplored-state generalization failure.

Keywords: length generalization, RNNs, representational drift, formal languages

1. Introduction

Understanding how and why recurrent sequence models fail to generalize beyond the training distribution is a central question in machine learning (Lake and Baroni, 2018; Delétang et al., 2022). The unexplored-state hypothesis proposes that length-generalization failure occurs when long recurrent rollouts reach regions of hidden-state space not encountered during training, causing the learned readout to fail (Ruiz and Gu, 2025). The hypothesis thus characterizes length-generalization failure as a distribution shift in recurrent hidden states. Here, we identify a specific geometric mechanism through which such a shift can emerge.

We trained linear RNNs with nonlinear readouts to predict the next token in a bounded-depth Dyck language, a canonical formal-language setting for studying stack-based computation (Hewitt et al., 2020). These models are both analytically tractable and share similar architectural properties to state-space models (Smith et al., 2022) and deep LRUs (Orvieto et al., 2023). We identify a subset of these models which have both stable recurrent dynamics and highly linearly decodable representations of the task-relevant stack state yet nevertheless fail to generalize. The stack representations of these models undergo a coherent, low-dimensional drift over long sequences. We show that a targeted test-time intervention on the drift-associated dimension leads to length generalization without any retraining. These results identify a concrete mechanism by which task-relevant representations can enter unexplored regions of hidden-state space and lead to generalization failure.

© 2026 A. Mah & A.H. Williams.

Extended Abstract Track

2. Preliminaries

2.1. Dyck-(k,m) Words

A Dyck-k word is a well-nested sequence of brackets drawn from k matching bracket types, terminated by a special end-of-sequence token, ω (Figure 1A). For each prefix, $y_{1:t} = y_1 y_2 ... y_t$, define the depth at position t, denoted $d(y_{1:t})$, to be the number of unmatched open brackets. We then define the bounded-depth Dyck-(k, m) language as $\{y \in \text{Dyck-}k \mid d(y_{1:t}) \leq m \text{ for all } t \leq |y|\}$, i.e., the set of all Dyck-k words whose maximum depth does not exceed m.

Briefly, the next-token distribution depends on the ordered sequence of unmatched open brackets, called the stack, denoted q_t. There are a total of $|Q| = (k^{m+1} - 1)/(k - 1)$ total stack configurations including the empty stack. An example next-token distribution for Dyck-(2,4) is given in Figure 1B. Additional details can be found in section A.

2.2. Recurrent Models

We trained single-layer linear RNNs with a single hidden-layer MLP readout to estimate the next-token probability given a Dyck-(k, m) prefix presented token-wise, $\hat{\boldsymbol{p}}_t \in \mathbb{R}^{2k+2}$:

$$\mathbf{h}_t = \mathbf{A}\mathbf{h}_{t-1} + \mathbf{B}\boldsymbol{x}_t, \quad \hat{\mathbf{p}} = \text{softmax}(f_{MLP}(\mathbf{h}_t)).$$

Sequences less than the maximum length were padded with a null token (for a total of $2k + 2$ tokens). Models were trained to minimize the next-token categorical cross-entropy loss with padding tokens masked. Additional details can be found in subsection B.1. Given a prefix with true next token probability distribution $p_{t,i} = p(y_{t+1} = i \mid y_{1:t})$, we assess the models' performance with the positional KL-divergence: Positional $D_{\text{KL}}(t) = D_{\text{KL}}(\boldsymbol{p}_t \mid\mid \hat{\boldsymbol{p}}_t)$

3. Results

Linear RNNs differ substantially in their ability to generalize beyond the training horizon. Here, we focus on models trained on the Dyck-(2,4) language; results for larger k and m are provided in subsection C.2. We trained the models on a set of sequences with a maximum length of $T_{\text{train}} = 128$, and tested on a set of sequences with a maximum length of $100 \cdot T_{\text{train}}$. For $t \leq T_{\text{train}}$, the models accurately predicted the true next-token distribution on unseen test trials (median positional $D_{\text{KL}} = 5.24 \times 10^{-4}$; Figure 1E), with performance comparable to nonlinear GRU models (median $D_{\text{KL}} = 5.50 \times 10^{-4}$). However, beyond the training length distribution, the linear models had remarkable variance in their ability to maintain accurate predictions (Figure 1C-E). We classified a model as a generalizer if its median positional D_{KL} over $99 \cdot T_{\text{train}} < t < 100 \cdot T_{\text{train}}$ was less than 10^{-2}. By this criterion, 13 out of 51 models generalized.

We first asked whether these differences could be explained by the stability of the linear dynamics. Generalizing models tended to have lower recurrent spectral radii (median $\rho(A) = 0.9989$) compared to non-generalizing models (median $\rho(A) = 1.0010$). However, spectral stability was neither necessary nor sufficient for generalization: two generalizing models had $\rho(A) > 1$, while six non-generalizing models had $\rho(A) < 1$ (Figure 2A). Thus, recurrent stability does not fully account for the observed differences in generalization.

Extended Abstract Track

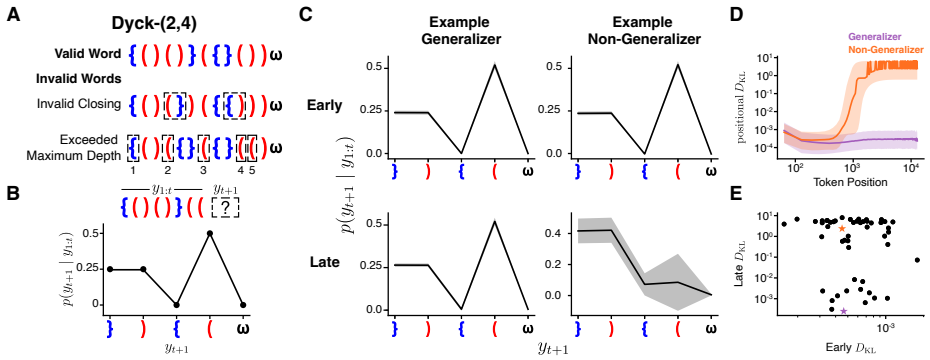


Figure 1: **A.** Example Dyck Words. **B.** Example next-token probability distribution. **C.** Example generalizing (left) and non-generalizing (right) models for early (top) and late (bottom) instances of a stack state. Data are mean ± standard deviation **D.** Positional D_{KL} for examples from C. Data are median and 25th and 75th quartiles. **E.** Early and late D_{KL} across independently trained models (stars indicate examples from C).

Models with unstable recurrent dynamics have a potential source of long-horizon failure, as small deviations along an unstable mode can be amplified over sufficiently long sequences. We therefore focused on the more surprising subset of six models that failed to generalize despite having stable recurrent dynamics. These stable non-generalizers provide a setting in which failure cannot be attributed to this simple dynamical instability.

Stable non-generalizers retain a linearly decodable representation of the stack. Correct next-token prediction requires the model to maintain and update the stack according to each new token. We therefore asked whether the stable, non-generalizing models failed to learn a valid stack representation and instead relied on some heuristic that was effective within the training length distribution, but failed beyond it. We trained multinomial logistic-regression probes on hidden states from a set of sequences spanning both early and late token positions and evaluated them on a separate set of held-out sequences.

Stack state was nearly perfectly decodable from generalizing models at both early and late positions (median accuracy = 0.997 and 1.000, respectively; Figure 2B). Across all non-generalizing models, decoding was substantially poorer, although still above chance (chance = $1/31 \approx 0.03$), with median early accuracy = 0.20, and median late accuracy = 0.24. However, all six stable non-generalizers retained high stack-decoding accuracy at long sequence lengths (median early accuracy = 0.916, median late accuracy = 0.995). Thus, these models failed behaviorally despite retaining a highly linearly decodable representation of the task-relevant stack state.

Stable non-generalizers exhibit low-dimensional representational drift. The preserved stack information suggests that failure may reflect not loss of representational content, but a change in where that information is represented (Figure 2C). For each stack state, we defined its representational displacement as the difference between its late- and early-position hidden-state centroids (subsection B.4). Singular value decomposition of the

Extended Abstract Track

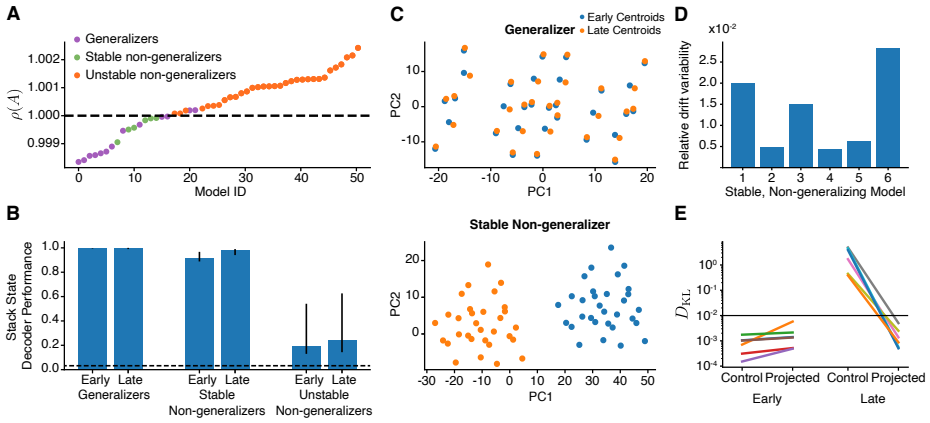


Figure 2: **A.** Recurrent weight spectral radii for each independently trained model. **B.** Linear decoding for early and late token positions by model category. Data are median with 25th and 75th percentiles. **C.** Top-2 PCA projection of stack state centroids for early and late token positions of example generalizing model (top) and stable, non-generalizing model (bottom). **D**. Relative variability of stack-state displacement along the leading drift axis for each stable non-generalizer. **E.** Drift-axis ablation performance at early (left) and late (right) token positions.

uncentered displacement matrix revealed that the displacement was predominantly one-dimensional for all stable non-generalizers, with the leading singular component accounting for more than 98% of the total squared displacement. Moreover, the displacement vectors were also coherently oriented and similar in magnitude. Across stack states, the standard deviation of their projection onto the leading axis was only $0.4\% - 2.8\%$ of the mean projection magnitude (Figure 2D). Thus, the change can be well approximated by a uniform translation of the stack representations along a single latent axis.

Finally, we tested whether activity along this drift-associated axis contributed causally to long-horizon failure. Ablating this single dimension dramatically improved long-horizon performance in all six stable non-generalizers (Figure 2E; subsection B.4). Median positional D_{KL} decreased from 2.79 before ablation to 1.12×10^{-3} after ablation, bringing performance close to that of naturally generalizing models (1.26×10^{-3}). In all six models, drift-axis ablation yielded lower long-horizon D_{KL} than any of 100 random one-dimensional ablations ($p_{\mathrm{emp}} = 1/101 < 0.01$ for all models; Figure 3). Importantly, the intervention did not disrupt performance within the training horizon (median early-position $D_{\mathrm{KL}} = 1.38 \times 10^{-3}$). Together, these results show that the dimension along which the stack representation drifts is specifically implicated in the loss of long-horizon performance.

4. Discussion

These results provide a concrete geometric instantiation of the unexplored-state hypothesis: the computational state remains linearly decodable, but its representation moves away from its training-regime location in hidden-state space, potentially placing it outside the region

Extended Abstract Track

on which the readout was trained. More broadly, our findings also suggest that global dynamical stability and representational stability are distinct properties. Constraining the recurrent dynamics to be globally stable does not necessarily ensure that task-relevant representations remain appropriately positioned over long horizons. This result may be relevant to state-space sequence models, which similarly combine linear recurrent dynamics with nonlinear readouts and often incorporate stability-oriented parameterizations. Relating the observed representational drift directly to the learned recurrent dynamics and testing whether explicitly constraining such drift during training improves length generalization are important next steps.

References

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, et al. Neural networks and the chomsky hierarchy. *arXiv preprint arXiv:2207.02098*, 2022.

John Hewitt, Michael Hahn, Surya Ganguli, Percy Liang, and Christopher D Manning. Rnns can generate bounded hierarchical languages with optimal memory. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1978–2010, 2020.

Patrick Kidger and Cristian Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR, 2018.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International conference on machine learning*, pages 26670–26698. PMLR, 2023.

Ricardo Buitrago Ruiz and Albert Gu. Understanding and improving length generalization in recurrent models. *arXiv preprint arXiv:2507.02782*, 2025.

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

Extended Abstract Track

Appendix A. Dyck Words

Standard Dyck-(k, m) words are generated autoregressively, with the distribution of the next token determined by the current depth according to the following rules:

1. At depth zero, the sequence terminates by ω with probability $1/2$; otherwise, an opening bracket is generated with probability $1/2$, chosen uniformly from the k bracket types.
2. At intermediate depths, $0 < d(y_{1:t}) < m$, the next token is either the closing bracket matching the most recently opened unmatched bracket, with probability $1/2$, or a new opening bracket, with probability $1/2$, again chosen uniformly from the k types.
3. At the maximum depth m, no additional bracket may be opened, so the most recently opened unmatched bracket is closed with probability 1.

This generative process results in an exponential distribution of sequence lengths. To ensure that the models saw enough examples of long sequences during training, we used a modified algorithm where each sequence had a minimum length drawn from a power-law distribution with $\alpha = 0.9$ with a maximum length of $T_{\text{train}} = 4m^2 + 16m$ (maximum length taken from Hewitt et al., 2020). When a prefix was shorter than its minimum length, the probability of emitting ω was set to 0. After a sequence reached its minimum length, the sequence was generated according to the algorithm above. This ensured that the training sequence distribution was at least as heavy-tailed as the minimum length distribution. Here, we trained on $(k, m) = (2, 4), (4, 4), (8, 4)$ which have $|Q| = \{31, 341, 4681\}$, respectively.

Appendix B. Methods

B.1. Model Architecture

We trained linear RNNs of the form

$$\begin{aligned} \mathbf{h}_t &= \mathbf{A}\mathbf{h}_{t-1} + \mathbf{B}\boldsymbol{x}_t \\ \hat{\mathbf{p}} &= \text{softmax}(f_{MLP}(\mathbf{h}_t)). \end{aligned}$$

The model components were initialized as follows:

1. Recurrent weights: $\mathbf{A} \in \mathbb{R}^{n \times n}$ was initialized as an orthogonal matrix scaled by $1/n$ where $n = 6m \log_2 k$ (adapted from Hewitt et al., 2020).
2. Input embedding: $\mathbf{B} \in \mathbb{R}^{n \times (2k+2)}$ was initialized row-wise from a standard Gaussian distribution and scaled to length λ which depended on (k, m) and was chosen via a hyperparameter grid search using the final validation loss.
3. The readout MLP f_{MLP} had one hidden layer with a soft-plus nonlinearity and layer norm applied to the final layer, where each weight matrix was initialized row-wise from a standard Gaussian distribution and scaled to unit length.

Models were implemented using the Equinox library (Kidger and Garcia, 2021). GRU baselines were initialized using the default Equinox initialization (Kidger and Garcia, 2021) with the same number of hidden-units as the linear models.

Extended Abstract Track

B.2. Model training

Training was performed using the Adam optimizer from the Optax library. RNNs were initialized with a unique random seed. Training began with a learning rate of 0.001. Training epochs were divided into 8192 mini-batches of 128 sequences. Sequences were randomly generated for each epoch. Models were updated according to the mean categorical cross-entropy loss after each mini-batch, with padding tokens masked.

After each epoch, the validation loss was computed on a static set of 32,768 sequences with the same maximum length as the training set. If the validation loss failed to improve upon the previous minimum, the learning rate for the subsequent epoch was reduced by half, otherwise it remained the same. Training was terminated once the validation loss failed to improve for three consecutive epochs. Models were fit on a NVIDIA RTX A6000 (48 GB VRAM, CUDA 12.8, driver 570.172.08). Each training run required between 15 and 30 mins on a single RTX A6000 GPU.

All analyses here were performed on an unseen test set of 5,000 independently sampled sequences according to the same power-law ($\alpha = 0.9$) as the training and validation set, with a maximum length of $100 \cdot T_{\text{train}} = 12,800$, chosen to test for robust length generalization.

B.3. Linear Decoding

We trained multinomial logistic decoders using the standard scikit-learn implementation with a standard scaler (mean 0, unit variance). The training set consisted of hidden activations randomly sampled across valid token positions from 400 sequences (total of 994,823 token positions). The testing set consisted of 100 unseen sequences (total of 236,022 token positions). We further divided the test set into early ($t < T_{\text{train}}$) and late ($90T_{\text{train}} < t < 100T_{\text{train}}$). Note that the definition of late differs from the behavioral definition in order to ensure sufficient late trials in the smaller test set.

B.4. Drift Analyses

For each model, we calculated the centroid of the hidden states associated with each stack configuration, q, for early and late positions as

$$\boldsymbol{\mu}_q^{\text{early}} = \mathbb{E}[\boldsymbol{h}_t \mid q_t = q, t \leq t_{\text{train}}]$$

and

$$\boldsymbol{\mu}_q^{\text{late}} = \mathbb{E}[\boldsymbol{h}_t \mid q_t = q, t > N \cdot t_{\text{train}}],$$

where N depends on (k, m) to ensure enough stack states q were visited, and may differ from the definition of late for behavioral results. We then defined the centroid displacement as

$$\boldsymbol{\Delta}_q = \boldsymbol{\mu}_q^{\text{late}} - \boldsymbol{\mu}_q^{\text{early}}$$

and the full displacement matrix as

$$\mathbf{D} = [\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_2, \ldots, \boldsymbol{\Delta}_{|Q|}].$$

If a stack state was not visited, that column was removed from $\mathbf{D}$. $N = 98$ for Dyck-(2,4) and Dyck-(4,4), and $N = 90$ for Dyck-(8,4).

We then computed the singular value decomposition of the full displacement matrix,

$$\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top.$$

Using the cumulative sum of squared singular values,

$$s_k = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_i \sigma_i^2},$$

we defined the effective dimensionality, d, as

$$d = \min\{i \mid s_i > 0.9\}.$$

We called the leading left singular vector, $\boldsymbol{u}_1$, the drift axis. To quantify the direction and magnitude of drift for each stack state (columns of $\mathbf{D}$), we projected each displacement vector onto the drift axis with

$$\boldsymbol{\alpha} = \mathbf{D}^\top \boldsymbol{u}_1,$$

and compared both the sign and magnitude of each element of $\boldsymbol{\alpha}$. We report $\text{std}(\boldsymbol{\alpha})/|\,\text{mean}(\boldsymbol{\alpha})|$ for each stable, non-generalizing model.

B.5. Drift Ablation

To ablate activity of the model along the drift axis, we collected the full, unablated hidden activations for each sequence (appended end-to-end for each sequence), $\mathbf{H}$, and formed the ablated hidden activation matrix, $\hat{\mathbf{H}}$ as

$$\hat{\mathbf{H}} = (\mathbb{I}_n - \mathbf{u}_1\mathbf{u}_1^\top)\mathbf{H}.$$

Finally, we calculated the projected model's output by feeding $\hat{\mathbf{H}}$ through the original model's MLP readout and computed the positional D_{KL} using the projected readout. We computed a null distribution by first sampling a vector, $\tilde{\boldsymbol{v}}$, where $\tilde{v}_i \sim \mathcal{N}(0, 1)$, and projecting out activity along $\boldsymbol{v} = \tilde{\boldsymbol{v}}/||\tilde{\boldsymbol{v}}||$ for 100 independent samples $\boldsymbol{v}$ for each model.

Appendix C. Additional Results

C.1. Additional Dyck-(2,4) Results

As with the stable non-generalizing models, the displacement was roughly one-dimensional, with the leading singular component accounting for an average of 99.1% of the total squared displacement ($N = 32$ models). However, there was variability in the efficacy of the displacement dimension ablation experiment. Median positional D_{KL} decreased to below 10^{-2} for 12 of the 32 models (median $= 3.45 \times 10^{-3}$; Figure 4), while the other 20 models remained above 10^{-2} (median $= 1.36$).

Extended Abstract Track

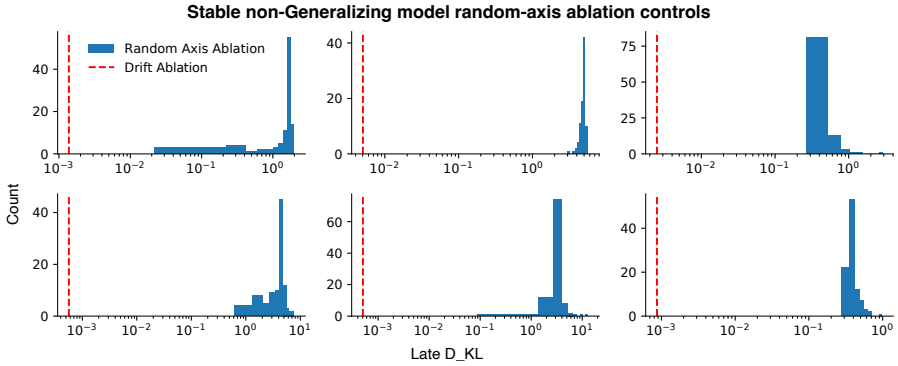


Figure 3: True drift ablation vs. 100 random-axis ablations for the 6 stable, non-generalizing models

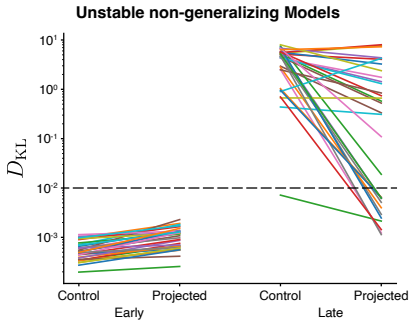


Figure 4: Drift axis ablation results for Dyck-(2,4) unstable non-generalizing models

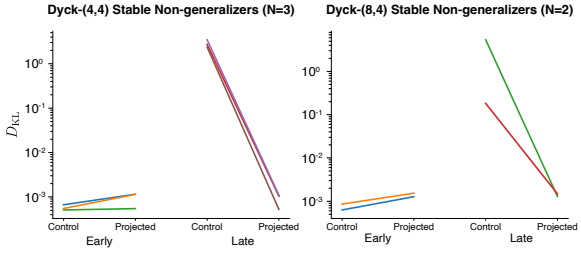


Figure 5: Drift ablation results for Dyck-(4,4) (left) and Dyck-(8,4) (right)

Extended Abstract Track

C.2. Results for Larger Dyck Languages

We also trained models on sequences generated from either Dyck-(4,4) or Dyck-(8,4) languages ($N = 10$ for each language). We find qualitatively similar results to Figure 2E for models trained on Dyck-$(4, 4)$ or Dyck-$(8, 4)$ languages. Namely, the stack representations of stable non-generalizing models ($N = 3$ for Dyck-(4,4) and $N = 2$ for Dyck-(8,4)) undergo a roughly 1-dimensional translation through hidden state space, and projecting out this dimension reduced late D_{KL} (Figure 5).

Extended Abstract Track