Extended Abstract Track

Grokking in recurrent networks with attractive and oscillatory dynamics

Keith T. Murray KTMURRAY@MIT.EDU
Massachusetts Institute of Technology, Cambridge, MA 02139

Editors: Sophia Sanborn, Christian Shewmake, Simone Azeglio, Nina Miolane

Abstract

Generalization is perhaps the most salient property of biological intelligence. In the context of artificial neural networks (ANNs), generalization has been studied through investigating the recently-discovered phenomenon of "grokking" whereby small transformers generalize on modular arithmetic tasks. We extend this line of work to continuous time recurrent neural networks (CT-RNNs) to investigate generalization in neural systems. Inspired by the card game SET, we reformulated previous modular arithmetic tasks as a binary classification task to elicit interpretable CT-RNN dynamics. We found that CT-RNNs learned one of two dynamical mechanisms characterized by either attractive or oscillatory dynamics. Notably, both of these mechanisms displayed strong parallels to deterministic finite automata (DFA). In our grokking experiments, we found that attractive dynamics generalize more frequently in training regimes with few withheld data points while oscillatory dynamics generalize more frequently in training regimes with many withheld data points.
Keywords: Generalization, Dynamical Systems, Representation, Interpretability

1. Introduction

Generalization is essential for biological intelligence, enabling animals and humans to extrapolate learned knowledge to new situations, thereby enhancing adaptability and survival. Given the numerous ecologically-relevant tasks biological agents perform, it comes as no surprise that the mechanisms through which neural systems generalize remain elusive. In artificial neural networks (ANN), "grokking" has emerged as a compelling phenomenon to study generalization (Power et al., 2022). This term refers to the ability of ANNs, particularly small transformers, to generalize learning from a limited input space to the entire input space in algorithmic tasks such as modular arithmetic.

To advance our understanding of generalization in biological agents, we sought to investigate the grokking phenomenon in continuous time recurrent neural networks (CT-RNNs). Our choice to investigate CT-RNNs was motivated by their utility in modeling neural systems (Sussillo, 2014; Barak, 2017). CT-RNNs have not only been experimentally validated to reproduce neural dynamics observed in primates (Mante et al., 2013; Chaisangmongkon et al., 2017; Sohn et al., 2019), but have also suggested a series of curious links between neural, dynamical, and computational systems (Sussillo and Barak, 2013; Yang et al., 2019; Vyas et al., 2020; Driscoll et al., 2022).

In this work, we show that CT-RNNs trained on our novel modular arithmetic task can learn two distinct dynamical mechanisms. The first mechanism is characterized by attractive dynamics and represents inputs as vectors. The second mechanism is characterized by oscillatory dynamics and represents inputs as angles. Intriguingly, both mechanisms are in

differing ways analogous to deterministic finite automata (DFA). In our grokking experiments, we found that both mechanisms achieved comparable rates of generalization, with attractive dynamics exhibiting a significant advantage in low-generalization data regimes, and oscillatory dynamics exhibiting a slight advantage in high-generalization data regimes.

2. Modular Arithmetic Task

The original modular arithmetic task used to elicit the grokking phenomena took the following form (Power et al., 2022):

$$a + b \equiv c \pmod{p} \tag{1}$$

Here, p is a preset prime number and $a, b, c \in \{0, 1, \ldots, p-1\}$. The task consists of summing inputs a and b and computing the residue $c \pmod{p}$. To elicit grokking, ANNs were trained on 30% of possible a, b pairs and generalized to compute the 70% of a, b pairs held-out.

Inspired by the card game SET (McMahon et al., 2017), we formulated the modular arithmetic task as follows to elicit interpretable CT-RNN dynamics:

$$a + b + c \equiv 0 \pmod{3} \tag{2}$$

Here, $a, b, c \in \{0, 1, 2\}$ and the task now consists of classifying if the congruence relation is valid (e.g. $0+1+2 \equiv 0 \pmod 3$ and $0+1+1 \not\equiv 0 \pmod 3$). Notice how this task essentially asks if a, b, c are all the same or all different. This is the primary rule in the card game SET, and in this work, we will refer to $a, b, c \in \{0, 1, 2\}$ with colors: $a, b, c \in \{\textit{green}, \textit{purple}, \textit{red}\}$. Task implementation details are described in Appendix A.

3. Dynamical Mechanisms

We observed that a CT-RNN can learn two distinct mechanisms when trained on our modular arithmetic task: one characterized by attractive dynamics and the other characterized by oscillatory dynamics. CT-RNN implementation details are described in Appendix B.

3.1. Attractive Dynamics

In a CT-RNN employing attractive dynamics, inputs are represented as vectors. As these vectors are injected into the network, they drive the network activity between various fixed-point attractors within the network's phase space (Figure 1 Top Row). Crucially, the vector representation of each input maintains a consistent direction throughout the phase space.

This mechanism draws parallels to a DFA, where symbols are sequentially processed, influencing the current state within a graph based on the specific symbol read. Here, fixed-point attractors in CT-RNNs parallel states in DFA, residue inputs parallel input symbols, and the ensuing vector shifts parallel state transitions.

This mechanism's emergence is conditioned on setting the time constant, τ, to 100ms, aligning with previous CT-RNN works (Chaisangmongkon et al., 2017; Yang et al., 2019). The configuration of τ is critical, impacting the representation and processing of inputs, and hence, the overall behavior of CT-RNNs.

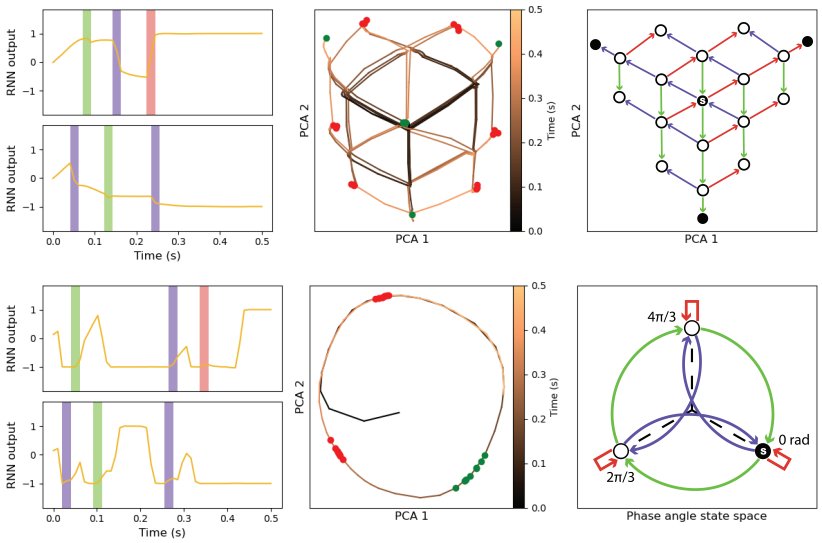

Figure 1: **(Top Row)** Example outputs, PCA embedding, and DFA for the attractive mechanism. **(Bottom Row)** Example outputs, PCA embedding, and DFA for the oscillatory mechanism.

3.2. Oscillatory Dynamics

In a CT-RNN employing oscillatory dynamics, inputs are represented as angles, and the intrinsic dynamics are characterized by a limit cycle (Strogatz, 1994). In essence, these inputted angles shift the phase angle of the intrinsic limit cycle (Figure 1 Bottom Row). Further mechanistic details are described in Appendix C and in Murray (2023).

Strikingly, this mechanism preserves the previously outlined analogy to DFA, where DFA states are now parallel to unique oscillation phases of the limit cycle. Given that a more compact DFA can arise from the networks' representing inputs as angles instead of as vectors, we hypothesized that the oscillatory mechanism would be more apt for generalization on our modular arithmetic task compared to the attractive mechanism.

The emergence of this oscillatory mechanism is conditioned on setting τ to 10ms. Previous CT-RNN works have noted the emergence of oscillatory dynamics; however, the associated time constants were considerably greater than 10ms (Kay et al., 2022; Pals et al., 2023). Remarkably, this mechanism bears a resemblance to oscillatory models of grid cells hypothesized to exist at the cellular level (Burgess et al., 2007), potentially mirrored more closely by setting τ to 10ms.

4. Grokking Experiment

To assess each dynamical mechanism's ability to generalize, we trained CT-RNNs with various congruence relations excluded from the training set. We found that both mechanisms are capable of generalizing to a range of excluded congruence relations; however, not all CT-

RNNs fully generalized, an issue we speculate arose from *exploding and vanishing gradients* (Pascanu et al., 2012; Park et al., 2023). To determine the probability of each dynamical mechanism successfully generalizing, we trained approximately 20,000 CT-RNNs under a variety of experimental conditions. Experimental details are described in Appendix D.

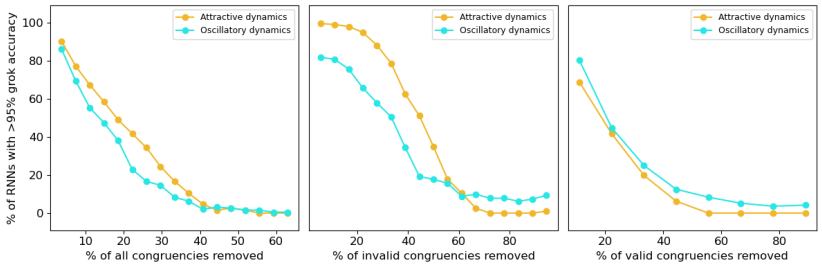

Figure 2: Rates of generalization under a variety of experimental conditions.

We initially expected that oscillatory dynamics would achieve higher rates of generalization due to the compact nature of its DFA representation; however, results were mixed. While attractive dynamics tended to outperform oscillatory dynamics with few congruence relations excluded, oscillatory dynamics were able to achieve non-zero generalization rates with many congruence relations excluded (Figure 2).

5. Discussion

In this work, we took inspiration from the card game SET to reformulate previous modular arithmetic tasks and showed that CT-RNNs can learn two distinct dynamical mechanisms: one characterized by attractive dynamics and the other characterized by oscillatory dynamics. We further showed how attractive dynamics generalize more frequently in low-generalization data regimes while oscillatory dynamics generalize more frequently in high-generalization data regimes.

The design of our novel modular arithmetic task not only drew inspiration from previous grokking tasks (Power et al., 2022) but also from the "3-bit flip-flop task" (Sussillo and Barak, 2013; Kim et al., 2023). We hope that our task will inspire the design of future tasks that seek to elucidate the dynamical, geometric, and computational properties of CT-RNNs.

While the analogy between RNNs and DFA has been previously investigated (Cleeremans et al., 1989), we speculate that developing this analogy further could enhance the interpretability of RNNs. Furthermore, this analogy echos work on understanding the internal reasoning mechanisms of shallow transformers as being comprised of semiautomata (Liu et al., 2022).

Future research could benefit from examining how different initializations of recurrent weights influence the speed and extent of generalization in CT-RNNs implementing either dynamical mechanism. Previous observations suggest that certain initialization strategies may improve performance in RNNs characterized by oscillatory dynamics (Park et al., 2023), indicating that optimized initializations could enhance generalization rates.

Additionally, investigating the rate at which CT-RNNs generalize during training is another avenue for future research. While Power et al. (2022) found that transformers continued to generalize on modular arithmetic tasks well past the point of overfitting, our work did not investigate if CT-RNNs display similar training behavior.

Acknowledgments

We thank Nancy Lynch, Brabeeba Wang, Sabrina Drammis, Nikasha Patel, Josh Liu, and Julian Liu for helpful discussions throughout the conceptualization, development, and execution of this project. We also thank the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing HPC resources that have contributed to the research results reported within this project.

References

Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Current opinion in neurobiology*, 46:1–6, 2017.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Neil Burgess, Caswell Barry, and John O'Keefe. An oscillatory interference model of grid cell firing. *Hippocampus*, 17(9):801–812, 2007.

Warasinee Chaisangmongkon, Sruthi K Swaminathan, David J Freedman, and Xiao-Jing Wang. Computing by robust transience: How the fronto-parietal network performs sequential, category-based decisions. *Neuron*, 93(6):1504–1517, 2017.

Axel Cleeremans, David Servan-Schreiber, and James L McClelland. Finite state automata and simple recurrent networks. *Neural computation*, 1(3):372–381, 1989.

Christopher J Cueva and Xue-Xin Wei. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *arXiv preprint arXiv:1803.07770*, 2018.

Laura Driscoll, Krishna Shenoy, and David Sussillo. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *bioRxiv*, 2022.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2023. URL http://github.com/google/flax.

Kenneth Kay, Xue-Xin Wei, Ramin Khajeh, Manuel Beiran, Christopher J Cueva, Greg Jensen, Vincent P Ferrera, and LF Abbott. Neural dynamics and geometry for transitive inference. *bioRxiv*, 2022.

Timothy Doyeon Kim, Tankut Can, and Kamesh Krishnamurthy. Trainability, expressivity and interpretability in gated neural odes. *arXiv preprint arXiv:2307.06398*, 2023.

Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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.

Liz McMahon, Gary Gordon, Hannah Gordon, and Rebecca Gordon. *The Joy of SET: The Many Mathematical Dimensions of a Seemingly Simple Card Game*. Princeton University Press, Princeton, NJ, 2017.

Keith T Murray. Recurrent networks recognize patterns with low-dimensional oscillations. *arXiv preprint arXiv:2310.07908*, 2023.

Matthijs Pals, Jakob H Macke, and Omri Barak. Trained recurrent neural networks develop phase-locked limit cycles in a working memory task. *bioRxiv*, 2023.

Il Memming Park, Ábel Ságodi, and Piotr Aleksander Sokół. Persistent learning signals and working memory without continuous attractors. *arXiv preprint arXiv:2308.12585*, 2023.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Albert Reuther, Jeremy Kepner, Chansup Byun, Siddharth Samsi, William Arcand, David Bestor, Bill Bergeron, Vijay Gadepally, Michael Houle, Matthew Hubbell, Michael Jones, Anna Klein, Lauren Milechin, Julia Mullen, Andrew Prout, Antonio Rosa, Charles Yee, and Peter Michaleas. Interactive supercomputing on 40,000 cores for machine learning and data analysis. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pages 1–6. IEEE, 2018.

Hansem Sohn, Devika Narain, Nicolas Meirhaeghe, and Mehrdad Jazayeri. Bayesian computation through cortical latent dynamics. *Neuron*, 103(5):934–947, 2019.

Steven H Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, And Engineering*. Westview Press, Boulder, CO, 1994.

David Sussillo. Neural circuits as computational dynamical systems. *Current opinion in neurobiology*, 25:156–163, 2014.

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.

David Sussillo, Mark M Churchland, Matthew T Kaufman, and Krishna V Shenoy. A neural network that finds a naturalistic solution for the production of muscle activity. *Nature neuroscience*, 18(7):1025–1033, 2015.

Saurabh Vyas, Matthew D Golub, David Sussillo, and Krishna V Shenoy. Computation through neural population dynamics. *Annual review of neuroscience*, 43:249–275, 2020.

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.

Appendix A. Modular Arithmetic Task Details

For our modular arithmetic task, the inputs, which can be denoted as $a, b, c \in \{0, 1, 2\}$, were embedded within a 100-dimensional vector. Each entry of this vector was sampled from a normal distribution, $\mathcal{N}(0, 1)$. Within a given training instance, the embedding vectors remained consistent; however, they varied across different training instances.

A single instance of our task, referred to as a trial, consisted of 50 time steps. During this trial, three embedded inputs were presented. Each input lasted for a span of two time steps, followed by a gap where no inputs would be presented for three consecutive time steps. In the absence of any input, a 100-dimensional zero vector was presented to the network. Notably, the initial three time steps and the final 10 time steps of the trial did not feature any input presentation. The time steps eligible for input presentation were selected uniformly at random.

Appendix B. CT-RNN Details

Our CT-RNN implementation took the following form:

$$\tau \dot{x}_i(t) = -x_i(t) + \sum_{k=1}^{N} J_{ik} r_k(t) + \sum_{k=1}^{N^{\text{in}}} B_{ik} u_k(t) + b_i + \eta_i(t) \tag{3}$$

$$r_i(t) = \tanh(x_i(t)) \tag{4}$$

$$z(t) = \sum_{k=1}^{N} W_k r_k(t) + b_{\text{out}} \tag{5}$$

Equation 3 was discretized using Euler's method and took the following form:

$$x_i(t+1) = (1 - \frac{\Delta t}{\tau})x_i(t) + \frac{\Delta t}{\tau}(\sum_{k=1}^{N} J_{ik} r_k(t) + \sum_{k=1}^{N^{\text{in}}} B_{ik} u_k(t) + b_i + \eta_i(t)) \tag{6}$$

Given that the task was a binary classification task, we used a mean squared error (MSE) loss function to evaluate a CT-RNNs performance on our task. We added an L2 regularization onto the rates of the CT-RNN (equation 4) which has been shown to increase interpretability (Sussillo et al., 2015; Cueva and Wei, 2018). The loss function took the following form:

$$\mathcal{L}(\hat{z}, z, \mathbf{r}) = \frac{1}{5} \sum_{t=T-5}^{T} (\hat{z} - z(t))^2 + \frac{\lambda}{TN} \sum_{t,i=1}^{T,N} r_i^2(t) \Delta t \tag{7}$$

All trainable parameters $(\boldsymbol{J}, \boldsymbol{B}, \boldsymbol{b}, \boldsymbol{W}, b_{\text{out}})$ were updated using the AdamW learning algorithm (Loshchilov and Hutter, 2017). The learning rate was set to 10^{-3} and the weight decay was set to 10^{-4}. Trainable parameters $\boldsymbol{J}, \boldsymbol{B}, \boldsymbol{W}$ were initialized using the Glorot uniform initializer with the scale set to 1.0 and the distribution as uniform (Glorot and Bengio, 2010). Trainable parameters $\boldsymbol{b}, b_{\text{out}}$ were initialized as zero vectors.

Table 1: Model parameters, values, and descriptions

Parameter	**Value**	**Description**
τ	10ms or 100ms	Time constant
Δt	10ms	Step size used for Euler's method
N	100	Number of CT-RNN neurons
N^{in}	100	Input dimension
$\eta_i(t)$	$\mathcal{N}(0, 0.10)$	ith-neuron noise at time t
$\hat{z}$	-1 or 1	Congruence relation label
T	50	Number of time steps per trial
λ	10^{-4}	L2 regularization

All code was implemented using the Jax ecosystem (Bradbury et al., 2018) and Flax library (Heek et al., 2023). All code was executed using the MIT Supercloud HPC (Reuther et al., 2018).

Appendix C. Oscillatory Mechanism Details

The oscillatory mechanism in a CT-RNN translates the incoming input into phase shifts that alter the network's oscillatory behavior. Consider a standard cosine function representing the network's intrinsic dynamics:

$$g(t) = \cos(wt) \tag{8}$$

The oscillatory mechanism modifies equation 8 to:

$$f(t) = \cos\left(wt + \int_0^t \phi(t')dt'\right) \tag{9}$$

$\phi(t')$ denotes the phase shift at time t', derived from the external input:

$$\phi(t) = \begin{cases} +\frac{2\pi}{3} & \text{if green at } t \\ -\frac{2\pi}{3} & \text{if purple at } t \\ 0 & \text{if red at } t \end{cases}$$

Equations 8 and 9 are plotted in Figure 3.

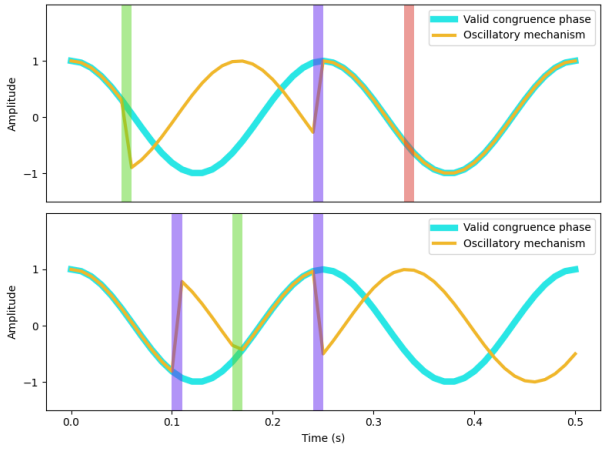

Figure 3: **(Top Row)** Oscillatory mechanism for a valid congruence relation. **(Bottom Row)** Oscillatory mechanism for an invalid congruence relation.

Appendix D. Grokking Experiment Details

Our grokking experiments were structured around three different subsets of congruence relations removed from the training set: (1) both valid and invalid congruence relations were excluded, (2) only valid congruence relations were excluded, and (3) only invalid congruence relations were excluded. Within each group, congruence relations were removed uniformly and without replacement to ensure unbiased sampling.

Each CT-RNN was trained for 5000 epochs and with mini-batches of 256 trails. For each congruence relation, there were at least 50 trials included. For each training dataset, both types of congruence relations had approximately the same number of trials. For each instance of excluded congruence relations, we trained a total of 384 CT-RNNs. These were equally divided based on their time constants: half (192 CT-RNNs) had $\tau = 10$ms and the remaining half had $\tau = 100$ms.

To estimate the probability of generalizing, we calculated the proportion of networks that achieved a high performance on held-out congruence relations. For each experimental condition and τ value, we determined the number of CT-RNNs achieving a validation accuracy of 95% or higher and divided it by 192, the total number of CT-RNNs for that specific experimental condition.