Relating Generalization in Deep Neural Networks to Sensitivity of Discrete Dynamical Systems

Jan Disselhoff
jadissel@uni-mainz.de

Michael Wand
wandm@uni-mainz.de

Abstract

The ability of deep neural networks to generalize over a large diversity of function modeling tasks, remains a key mystery of the field. While the workings of the network are fully known, it remains unclear which specific properties are necessary and/or sufficient for the observed generalization. In this paper, we approach the characterization of this generalization by studying the ability to learn the evolution of discrete dynamical systems. Our findings reveal a strong correlation between the number of examples needed for generalization and the sensitivity of the dynamical systems to perturbations of the initial state.

1 Introduction

In recent years, it has become clear that deep learning architectures can model a large set of naturally occurring systems. However, basic machine learning theory tells us that there have to be limits, and only systems aligned with the inductive bias of the network are *efficiently* learnable, i.e., learnable from realistic amounts of data. In order to understand this inductive bias better, we study learning of dynamical systems – based on the premise that natural processes follow physical dynamics. [Lin et al., 2017]

In order to experimentally investigate *which* dynamical systems are hard to learn, we choose a limited but expressive subset of discrete dynamical systems to model. To align with physical models, we limit ourselves to *local,time-invariant and translation-invariant* systems. Together with discreteness, these conditions coincide with the set of *Cellular Automata (CA)* [Hedlund, 1969].

Fully investigating CAs with a rather small number of inputs is already impractical, as the required space for encoding a single rule rises exponentially with the number of input parameters. On the other handy, specifying such rules implicitly introduces a large bias by favoring rules that are simple to encode implicitly. Instead we decided to investigate the subset of larger 2D CAs that are produced by repeated applications of smaller automata.

We explore under which conditions a fixed architecture will be able to easily learn to predict the k-step time evolution of such a CA, by training a large number of networks on different, randomly sampled rules. Our results suggest, that learnability correlates with the perturbation sensitivity of the function to be learned, i.e., the likelihood that small changes in the (2n+1)-by-(2n+1) input lead to changes in the output bit. This finding is a novel addition to previous studies on the network's perturbation sensitivity, the training process, or function class capacity characterizations such as through Rademacher or Lempel-Ziv complexity.

2 Background and Related Work

The exploration of the "prior of deep learning" has been multifaceted, encompassing various models and approaches to understand different aspects. The classical approach of infinite-width limit linearization, brought to light by [Neal, 1996, Jacot et al., 2018, Lee et al., 2018], has been instrumental

Machine Learning and the Physical Sciences Workshop, NeurIPS 2023.

in providing insights and revealing a Gaussian prior distribution in function space. Despite its foundational insights, particularly on generalization behavior [Belkin et al., 2019], the limitations [Roberts et al., 2022, Li et al., 2019] of this model accentuate the need for further exploration into the capabilities of deep networks.
Examining the intrinsic dimensionality and the complexity of data [Pope et al., 2021, Huh et al., 2023, Nakkiran et al., 2019] shows the effectiveness and learning constraints of neural networks, suggesting that neural networks tend to favor "simple" functions first, partially providing an explanation for their generalization performance on natural data.
The emergence of studies focusing on the artificial data-generating mechanisms [Pérez et al., 2018, Mingard et al., 2019, 2023, Bronstein et al., 2022] underscores the biases of neural networks towards simpler, low entropy Boolean functions, aligning with our observations on low perturbation sensitivity and learnability. Additionally, research utilizing 2D CAs as model systems [Gilpin, 2019, Springer and Kenyon, 2021, Lin et al., 2017] demonstrates the relationship between learnability, rule entropy, and the complexity emerging from iterated simple rules, providing complementary insights and highlighting unique learning challenges in different system models.

3 Methods

Let f_l be the local update rule for some (2n+1)-by-(2n+1) 2D Cellular automaton. f_l is then of the form $f_l : \{0,1\}^{(2n+1)\times(2n+1)} \to \{0,1\}$. Even for small values of n, arbitrary functions of this type are not learnable. For example, to specify a single function for $n = 3$, i.e. a 7-by-7 CA, we would need to specify the output for each single of the $2^{7\times7}$ input patterns, which would require 70TB of disk space – infeasible for systematic investigation using modern architectures.
For this reason we investigate only rules that can be defined as repeated applications of smaller 3-by-3 automata. While our problem is still complex, the underlying data generation process is simpler.

We separately investigate both random 3-by-3 automata and *outer-totalistic* (OT) automata. This subset, often called "life-like" automata, are not only translation invariant, but also *rotation* invariant. Further, there are "only" 262.144 different OT automata, making it possible to learn a representative fraction.

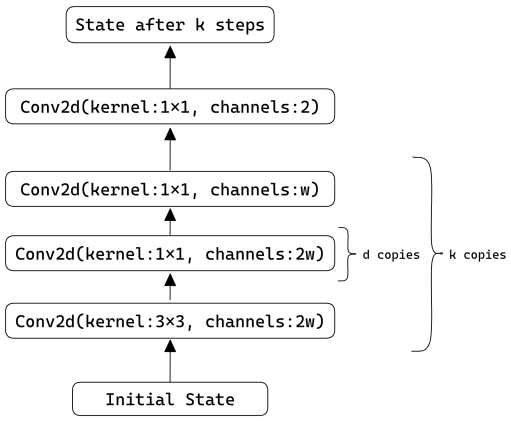
As we want to investigate learnability for repeated rules, we need to guarantee that our network is able to represent the provided functions. In order to do so, we design our architecture the following way: We create a module that is able to model a single iteration of any 3-by-3 CA, by using a 3-by-3 `Conv2D` filter, followed by several 1-by-1 `Conv2D` filters. All filters are separated by a `ReLU` function followed by `Batchnorm2D`. We then stack such modules, until there are as many as the number of iterations we want to predict. The modules do **not** share weights, as we do not want to bias the network towards learning a simple repeated rule. All our `Conv2D` filters use cyclic padding to ensure that the input and output shapes of the network are equivalent. For a graphical representation of our model, see Figure 1a.
Since each module is able to model a single application of a 3-by-3 rule, we know that the whole network is able to represent repeated applications of the rule. Note that this architecture is very similar to the ones used by Gilpin [2019] and Springer and Kenyon [2021], providing comparability with previous works in the literature.

We examine the ease with which such a network can learn arbitrary CA rules, a task which we find to depend strongly on a single value calculable from the transition function of the automaton:

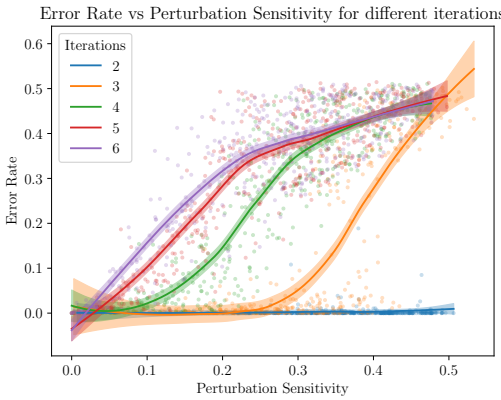
Definition 1 *Let f_l be a local transition map of a binary discrete dynamical system $f_l : \{0,1\}^n \to \{0,1\}$. Let $U(x)$ denote all immediate neighbors of x, i.e $U(x) = \{y \in \{0,1\}^n || x - y| = 1\}$. Perturbation Sensitivity (PS) is then defined as:*

$$PS(f) = \mathbb{E}_{x\in\{0,1\}^n}[\mathbb{E}_{y\in U(x)}|f(x) - f(y)|]$$

PS represents the probability that a specific output pixel will change if a random input pixel in the "perceptive field" is altered. It is a measure of unpredictability and can vary considerably between single and repeated applications of the same rule. PS is maximized at a value of 1 for the parity function, and minimized at 0 for constant functions. A random function (i.e. a function where neighboring inputs have no output correlation) has a PS of around 0.5.



(Figure 1a) Neural network architecture used in our experiments. Most experiments use $w = 256$ and $d = 4$. All filters are separated by a `ReLU` activation function, followed by a `BatchNorm2D` layer. Note that weights are **not** copied or shared.



(Figure 1b) Error rate for different training runs. Each dot represents training of a neural network to predict a random OT rule, given up to 2^{26} different input samples, colored by number of timesteps to predict. Higher perturbation sensitivity leads to an increase in error rate in each subexperiment. Solid lines represent a LOESS with 95% confidence Interval, calculated using scikit-misc.

4 Experiments and Results

To investigate how easily a given rule can be learned, we train a neural network to predict the result of applying that rule k times.

In Figure 1b we show how PS correlates to the error rate after training. While relatively noisy, we can see that an increase in PS is accompanied by an expected increase in the error rate. Further, we can see that our architecture is indeed strong enough to theoretically represent these functions, as rules with $k = 2$ converge to a very low error rate.
In order to see how PS influences behavior during training, we track the number of errors for new training samples. As each presented sample is a newly generated input from the data distribution, these values are equivalent to generalization performance at each training step. To reduce noise, we train 2880 different automata for $k = 2$ and $k = 3$.

For $k = 2$, we present the networks with 2048 batches, each consisting of 4 patterns of size 8×8, resulting in a maximum of 2^{19} different input-output pairs. For $k = 3$, we present the networks with 2048 batches, each consisting of 16 randomly sampled 32×32 patterns, resulting in a maximum of 2^{25} different input-output pairs. In both cases, the number of samples is significantly less than the theoretical number of distinct input patterns, but the number of gradient steps remains the same.
We observe that for $k = 2$ and $k = 3$ using our standard network architecture, PS strongly correlates with both speed of convergence and final performance. Rules with lower PS are easier to learn, while rules with higher sensitivity are harder, and with high enough sensitivity do not even converge to a good result during our training. The dependency between training speed and PS seems almost exponential: Trying to learn rules with a PS larger than ~ 0.3 quickly becomes much harder. This dependency exists both for $k = 2$ and $k = 3$.

It might be the case that these results are an artefact of choosing simple OT rules to generate data. We therefore repeat this experiment using non-OT rules. We again sample 2880 different rules using the sampling scheme described above. Results are similar, but slightly less smooth compared to OT automata. Increasing PS still increases error rate and far more samples are needed to obtain similar performance. We also compare our result to different architectures. If a single larger convolution is used instead of several 3-by-3 convolutions, performance suffers. While we still see a clear connection between PS and error rate during training, but in this case the error rate changes are less pronounced. Results for all experiments can be seen in Figure 2.

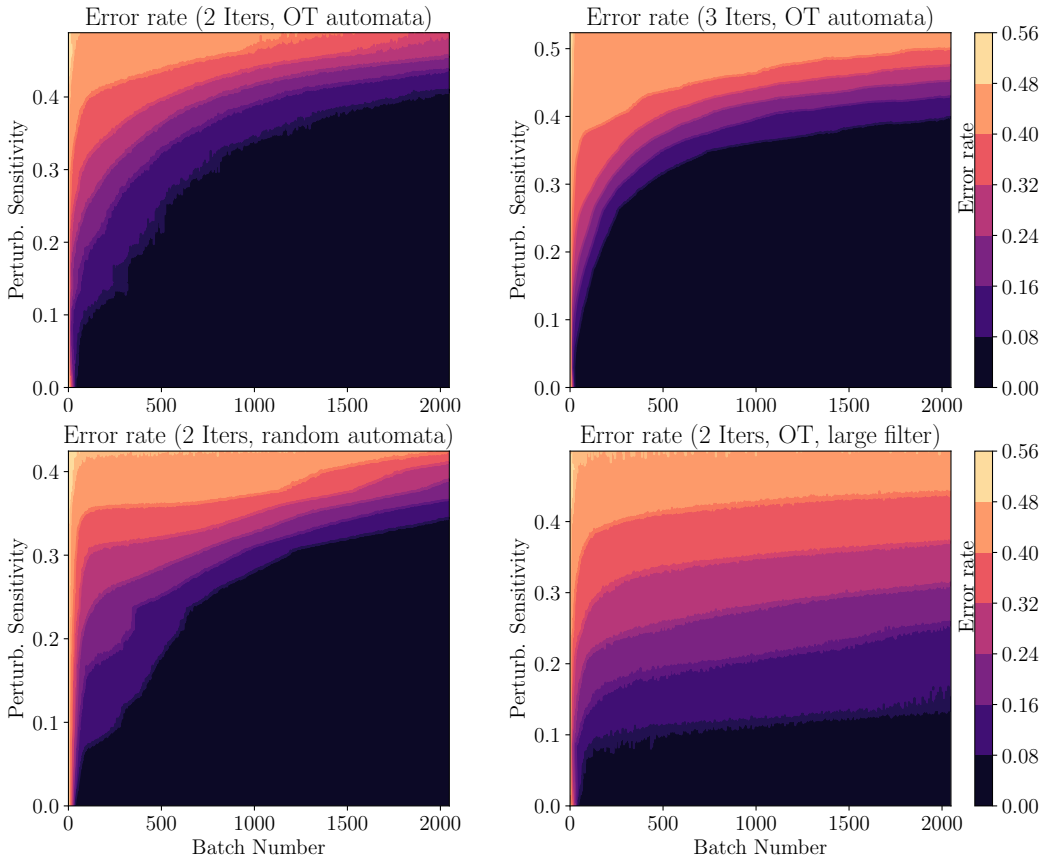


Figure 2: Evaluated performance during training. Plots show the mean behavior of rules with similar perturbation sensitivity. We display the evolution of the error rate for new batches during training. A clear correspondence between PS on the y-axis, and training behavior can be seen for each experiment. The number of samples needed to achieve similar generalization performance seems to rise exponentially with perturbation sensitivity, and a clear seperation in learnability is visible: Small changes in PS can have large impact on learnability. The only exception is the plot in the lower right corner, where we used a different architecture for comparison (One module with 5x5 conv filter, instead of two 3x3 modules).

5 Discussion

In this work we investigated the connection between learnability of repeated application of a discrete dynamical transition function – in this case provided by cellular automata – and a measure called Perturbation Sensitivity. We found that rules with lower Perturbation Sensitivity are in general easier to learn and generalize, with most experiments showing an exponential dependency between number of samples needed and Perturbation Sensitivity.

While this is a complementary result to previous findings on the inductive bias of deep networks, we consider this a first step towards characterizing the properties that make dynamical systems learnable and a better understanding of the generalization capabilities of deep neural networks.

6 Acknowledgements

This work has been supported by the Emergent AI Center, funded by the Carl-Zeiss-Stiftung.

References

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Ido Bronstein, Alon Brutzkus, and Amir Globerson. On the inductive bias of neural networks for learning read-once dnfs. In James Cussens and Kun Zhang, editors, *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pages 255–265. PMLR, 01–05 Aug 2022. URL `https://proceedings.mlr.press/v180/bronstein22a.html`.

William Gilpin. Cellular automata as convolutional neural networks. *Phys. Rev. E*, 100:032402, Sep 2019. doi: 10.1103/PhysRevE.100.032402. URL `https://link.aps.org/doi/10.1103/PhysRevE.100.032402`.

G. A. Hedlund. Endomorphisms and automorphisms of the shift dynamical system. *Mathematical systems theory*, 3:320–375, 1969.

Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks, 2023.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=B1EA-M-0Z`.

Zhiyuan Li, Ruosong Wang, Dingli Yu, Simon S Du, Wei Hu, Ruslan Salakhutdinov, and Sanjeev Arora. Enhanced convolutional neural tangent kernels. *arXiv preprint arXiv:1911.00809*, 2019.

Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168:1223–1247, 2017.

Chris Mingard, Joar Skalse, Guillermo Valle Pérez, David Martínez-Rubio, Vladimir Mikulik, and Ard A. Louis. Neural networks are a priori biased towards boolean functions with low entropy. *ArXiv*, abs/1909.11522, 2019.

Chris Mingard, Henry Rees, Guillermo Valle Pérez, and Ard A. Louis. Do deep neural networks have an inbuilt occam's razor? *ArXiv*, abs/2304.06670, 2023.

Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin Edelman, Fred Zhang, and Boaz Barak. Sgd on neural networks learns functions of increasing complexity. In *Neural Information Processing Systems*, 2019.

Radford M Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, NY, 1996.

Guillermo Valle Pérez, Chico Q. Camargo, and Ard A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. *ArXiv*, abs/1805.08522, 2018.

Phillip Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning, 2021.

Daniel A Roberts, Sho Yaida, and Boris Hanin. *The Principles of Deep Learning Theory*. Cambridge University Press Cambridge, MA, USA, 2022.

Jacob M. Springer and Garrett T. Kenyon. It's hard for neural networks to learn the game of life. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021. doi: 10.1109/IJCNN52387.2021.9534060.