Neural Lindenmayer Systems

Christoph W. Senn, Carmen Mei-Ling Frischknecht-Gruber[2], Mathias Weyland[2] and Rudolf M. Füchslin[2]

[1]Bandai Namco Studios, Japan
[2]Zurich University of Applied Sciences, Switzerland
chrigi.senn@gmail.com

Abstract

Our work introduces the Neural Lindenmayer system, an innovative approach to learning Lindenmayer rules from string representations in the presence of noise. It demonstrates its effectiveness in the example of the dragon curve. By integrating a selection network with multiple rule networks, our system effectively captures the variability in rule lengths. This offers a promising direction for analyzing and generating complex patterns observed in nature.

Introduction

Lindenmayer systems (L-systems) introduced in Lindenmayer (1968) are a class of string rewriting systems capable of describing a variety of systems. Learning the rule sets that describe a given system can thus give insights about its nature but also allows the generation of further samples.

Recent research has explored various computational strategies to decode and replicate the complex rule sets of L-systems, recognizing their potential and challenges. This has laid the groundwork for our Neural Lindenmayer system (NL-system) approach. Chen and Ross (2021), combined a convolutional neural network (CNN) and a genetic programming (GP) system to learn L-systems from tree images. The CNN was trained on renders of L-systems (trees and non-trees) and then used to estimate the fitness of population members of the GP system. Guo et al. (2020) used CNNs to extract the atomic structures, e.g., branches or leaves of a tree, and used this information to generate a n-ary tree. They then used this tree to infer a compact grammar that generates the given sample. Tu et al. (2023) used a differentiable renderer to learn iterative (affine) function systems from images of fractals and others. Although not L-systems in the strictest sense, they showed that gradient descent could be applied end-to-end to learn the rules behind such systems.

As an L-system consists, in general, of rewriting rules with various lengths, learning such a system end-to-end poses some difficulties. Our proposed NLS solves this issue by combining multiple rule networks, which learn rules of various lengths, with a selection network in a differential manner, as seen in Figure 1.

Neural Lindenmayer Systems

At the core of Neural Lindenmayer systems are a selection network θ_s and multiple rule networks $\theta_{1..N}$. The number of rule networks N and their respective lengths have to be predetermined as hyperparameters. Algorithm 1 describes how an output string is created from an input string s. By creating the output from a weighted sum of the rule network outputs, with weights based on the output of the selection network, we allow for the gradients to propagate through all networks.

Algorithm 1 Neural Lindenmayer system output creation

Require: string s, number of rule networks N, rule networks $\theta_{1..N}$, selection network θ_s, rule lengths $L_{1..N}$
  $outputs \leftarrow []$
  for all token t in s **do**
    $s \leftarrow softmax(select(t; \theta_s))$
    $idx \leftarrow argmax(s)$
    $x \leftarrow 0$
    for all $i = 1$ to N **do**
      $x \leftarrow x + rule(t; \theta_i) * s_i$
    end for
    $x \leftarrow x + floor(x) - detach\_grad(x)$
    $l \leftarrow L_{idx}$
    $outputs \leftarrow outputs + x_{1:l}$
  end for
  return $outputs$

Experiments

We trained our system on a dragon curve, specifically the Heighway Dragon as described by Gardner (1967), of up to ten iterations in a supervised manner. The input was the string representing a dragon curve for time step t, and the target string was the representation for timestep $t + 1$. Both strings were represented as integer vectors and the mean squared error (MSE) was used as a loss function for the network, which we trained using AdamWLoshchilov and Hutter (2017) optimizer. The number of rule networks and their

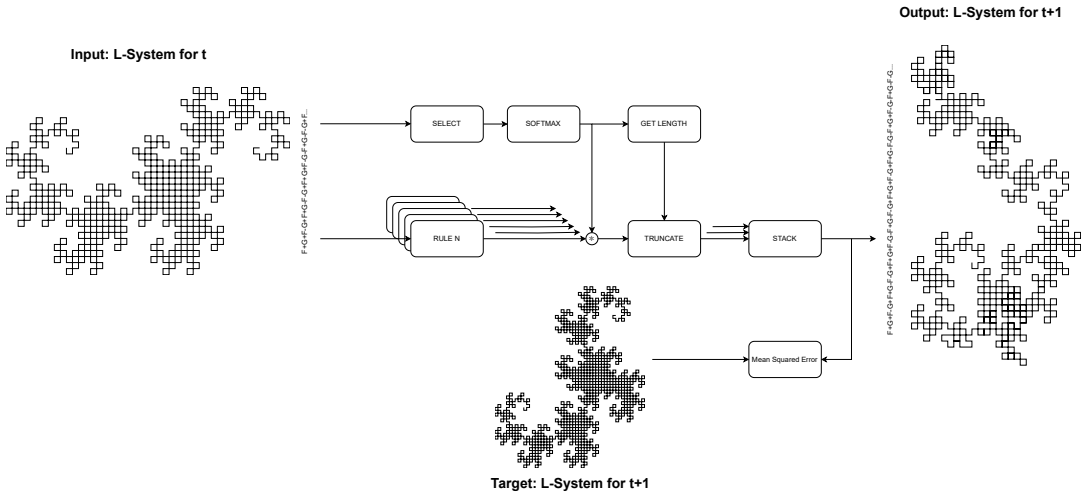

Figure 1: Architecture of our proposed "Neural Lindenmayer System". Tokenized strings are fed through a selection network and a series of rule networks. Their outputs are then combined using simple multiplication, enabling gradients to flow through the networks. To allow for rules with variable lengths, the output is truncated using the predefined length of the most likely rule.

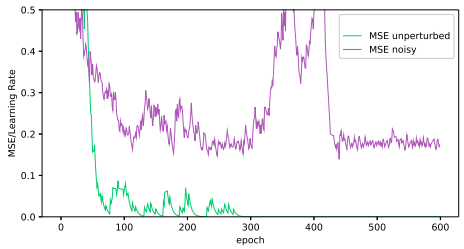

Figure 2: The MSE of the network during training for an unperturbed dragon curve (green) and a noisy dragon curve (purple). In the case of the unperturbed dataset, the error reaches 0 after a few epochs, indicating that the network learned to perfectly reproduce the given input.

lengths has to be determined in advance and has been chosen as $N = 2$ and $L_{1..N} = \{1, 3\}$ for the dragon curve. As noise often appears in biological systems, we also simulated mutations by randomizing the output of a rewriting step with probability $p = 0.01$ during the creation of the training dataset. This was done to examine if the system could still recover the original rules when shown mutated samples.

Results and Discussion

Our proposed system has demonstrated the ability to accurately learn and reproduce the underlying rules of the dragon curve under standard and noise-induced conditions. The MSE during training is shown in Figure 2. When training without added noise, the error quickly goes towards 0 indicating that the network reproduces the dragon curve perfectly. In the presence of noise, the error converges just below 0.2. This demonstrates the system's robustness against data imperfections and its potential for analyzing and synthesizing complex biological and fractal patterns where such noise is common. Due to the compact size of the rule networks used, explainable AI techniques, such as decision trees, provide a means for thorough analysis of the system's learned rules, thereby improving our comprehension and interpretability of the model.

The system's application to the dragon curve suggests that it can be adapted to more complex L-systems that simulate natural phenomena. Future efforts could explore its use in distinguishing between various L-system patterns and extending its learning capabilities to more diverse datasets.

Outlook and Conclusion

A possible direction for future work is to model multiclass L systems. i.e., rule systems that are conditioned on different curves can produce different outputs but also interpolate between them. This would enable further analysis of the relationships between similar L-systems.

Another direction involves learning rules from input-output combinations that span multiple iterations. Ideally, the underlying rules can be learned by presenting the network with only the final state of the system. For example, the network can learn to grow a tree by analysing its seed.

To conclude, we proposed a novel NLS that learns Lindenmayer rule systems from strings, even when noise is present in the training dataset. We hope that this work will inspire further investigations in the field of L-systems and their applications in nature.

References

Chen, X. E. and Ross, B. J. (2021). Deep neural network guided evolution of l-system trees. In *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE.

Gardner, M. (1967). Mathematical games. *Scientific American*, 216(3):124–129.

Guo, J., Jiang, H., Benes, B., Deussen, O., Zhang, X., Lischinski, D., and Huang, H. (2020). Inverse procedural modeling of branching structures by inferring l-systems. *ACM Transactions on Graphics*, 39(5):1–13.

Lindenmayer, A. (1968). Mathematical models for cellular interactions in development ii. simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology*, 18(3):300–315.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Tu, C.-H., Chen, H.-Y., Carlyn, D., and Chao, W.-L. (2023). Learning fractals by gradient descent. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(2):2456–2464.