Data-Centric Approach to Constrained Machine Learning: A Case Study on Conway's Game of Life

Anonymous submission

Abstract

This paper focuses on a data-centric approach to machine learning applications in the context of Conway's Game of Life. Specifically, we consider the task of training a minimal architecture network to learn the multi-step Game of Life, which is known to be challenging due to restrictions on the allowed number of trainable parameters. An extensive quantitative analysis showcases the benefits of utilizing a strategically designed training dataset, with its advantages persisting regardless of other parameters of the learning configuration (such as network initialization or optimization algorithm). Importantly, our findings highlight the integral role of domain expert insights in creating effective machine learning applications for constrained real-world scenarios.

Introduction

In this work we consider the problem of learning the rules of Conway's Game of Life, posed as an image-to-image translation task. Such a setting is inspired by (Springer and Kenyon 2021), where the authors investigate the ability of neural networks to learn the rules of Conway's Game of Life from the given state transition images. The authors have observed that such a task is often not achievable by conventional machine learning approaches and is only attainable under a sufficient network overparameterization (about $5 - 10$ times for a successful 1- or 2-step prediction). We restrict the setting by only working with minimally sufficient architectures and do not allow network overparameterization, which effectively puts us in a domain of *constrained machine learning*. To offset the lack of usual machine learning leniency, we allow for meticulous control of the training data. Specifically, we design a training board and compare the efficiency of training neural networks on this constructed board, rather than learning on randomly generated boards. We observe that even a single properly-crafted training board offers a significant increase in convergence rate and speed, especially on more challenging multi-step prediction tasks, regardless of the choice of a network initialization and an optimization algorithm. We use this observation to highlight the vital importance of the training data in constrained settings, which is well-aligned with the increasingly prominent concept of *data-centric machine learning*. We believe that this example can be easily translated to many real-world applications where machine learning approaches are restricted by the practical constraints like ethical, legal, security, or hardware limitations. Our contributions are the following:

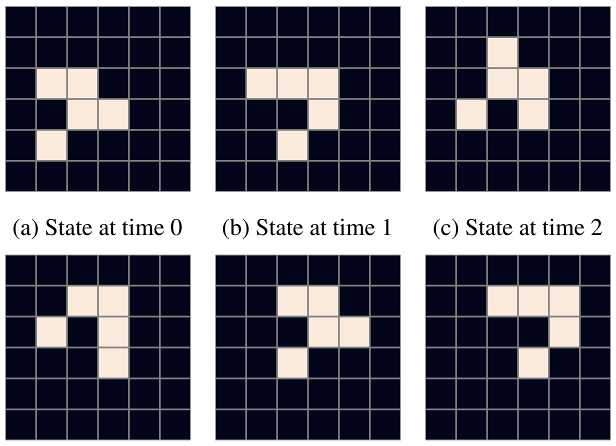


Figure 1: An example of a state trajectory in the Game of Life. Alive cells are white and dead cells are black.

- Develop multiple ways of representing the Game of Life as a neural network with minimal architecture;
- Explain the process of constructing the training data by analyzing the environment;
- Demonstrate the significance of the training data choice via an extensive qualitative analysis;
- Establish a simple yet challenging task that can be used by practitioners as a benchmark.

Conway's Game of Life. Game of Life is a classic cellular automaton devised by British mathematician John Conway in 1970. Operating on a two-dimensional grid of cells, each cell can exist in one of two states: alive or dead. The evolution of the system is solely determined by a set of rules that dictate the birth, survival, or death of cells based on their neighboring configurations, see Figure 1. Despite its apparent simplicity, Conway's Game of Life remains of significant interest in the fields of mathematics and machine learning due to its inherent complexity and some peculiar applications, see e.g. (Rennard 2002; Rendell 2011).

Conway's Game of Life serves as a computational model that explores how simple rules can lead to intricate and unpredictable patterns. This has led to various studies focusing on the mathematical properties of the Game of Life, including the classification and analysis of different types of structures and behaviors that emerge within the system (Adamatzky 2010; Hirte 2022). Moreover, it has been utilized as a platform for developing and testing various types of neural networks and deep learning architectures, with the goal of understanding how these models can learn to simulate and predict the evolution of the system, see e.g. (Krechetov 2021; Grattarola, Livi, and Alippi 2021).

In this work we explore the role of the training data on a task of learning the multi-step Game of Life with the minimal conventional architecture. Specifically, we use a convolutional neural network (CNN) to learn the rules of the game and predict the next state of the board. Training on our custom-made board outperforms the traditional paradigm in terms of accuracy and speed, and showcases the advantage of a data-centric approach in constrained machine learning applications.

Constrained Machine Learning. In recent years, machine learning has emerged as a transformative technology with applications spanning diverse domains, from natural language processing and computer vision to healthcare and finance. However, traditional machine learning algorithms often overlook the incorporation of critical constraints, which are paramount for ensuring ethical, reliable, and safe decision-making in real-world applications. Constrained machine learning has emerged as a powerful paradigm that addresses this pressing concern, encompassing a variety of techniques that integrate explicit constraints into the learning process, see e.g. (Perez et al. 2021; Yao et al. 2021; Gori, Betti, and Melacci 2023). These restrictions can be given as mathematical expressions, rules, or logical statements, and are typically derived from prior knowledge about the problem domain.

Constrained machine learning offers an immense potential in tackling complex challenges across various domains, such as healthcare (Ahmad, Eckert, and Teredesai 2018; Chen et al. 2021; Nguyen et al. 2021), autonomous vehicles (González et al. 2015; Dalal et al. 2018), finance (Bae et al. 2022; Al Janabi 2022), and natural language processing (Ammanabrolu and Hausknecht 2020; Yang et al. 2021; Zhang et al. 2022).

It is important to note that incorporating additional constraints can increase the implementation, training, and deployment complexity of machine learning models. But despite the potential challenges, constrained machine learning offers a promising avenue for improving the behavior and performance of machine learning models in critical real-life applications.

Data-Centric Machine Learning. Data-centric machine learning is an approach that prioritizes the quality, diversity, and relevance of data as a central focus in the training pipeline, as opposed to traditional machine learning where algorithms are often designed with a primary emphasis on model complexity and hyperparameter tuning. While these aspects remain important, data-centric machine learning recognizes that the quality and abundance of data available for training models can significantly impact their performance and generalization capabilities, see e.g. (Pan, Mason, and Matar 2022; Majeed and Hwang 2023). Data-centric machine learning finds application in numerous real-world scenarios across various industries, including healthcare (Zahid et al. 2021; Emmert-Streib and Yli-Harja 2022; Dritsas and Trigka 2022), finance (Liu et al. 2022; Horvatha, Gonzalezb, and Pakkanenc 2023), and environmental sciences (Devarajan et al. 2021; Li and Dong 2022).

Employing a data-centric approach poses several significant challenges that researchers and practitioners must address to achieve optimal results. Firstly, data quality and accessibility remain crucial hurdles, as obtaining large, diverse, and high-quality datasets can be costly and time-consuming (Roh, Heo, and Whang 2019). Dealing with imbalanced data, missing values, and noisy information requires careful preprocessing and augmentation techniques. Additionally, privacy and ethical concerns arise when handling sensitive data, demanding robust privacy preservation methods (Jo and Gebru 2020). Finally, domain expert knowledge is required to evaluate the data relevance and accuracy. These insights are essential for interpreting the data in the appropriate context and guiding the preprocessing steps, ensuring that the data is suitable for training and validating machine learning models effectively (Gennatas et al. 2020). Addressing these challenges is paramount to harnessing the full potential of data-centric machine learning in real-world applications. For a comprehensive overview of data-centric machine learning we refer the reader to the works (Anik and Bunt 2021; Miranda 2021; Seedat, Imrie, and van der Schaar 2022; Zha et al. 2023) and the references therein.

Related Work. We chose Game of Life as a challenging environment that is complicated for conventional machine learning algorithms to learn. Such a setting is inspired by (Springer and Kenyon 2021), where the authors investigated how network overparameterization affects performance. In contrast, we only consider the minimal network architectures.

Training networks of the smallest feasible architecture is considered in (Nye and Saxe 2018), where the authors observe the inability of learning the parity function and fast Fourier transform with the minimal networks. Research on the training of small networks is relevant, see e.g. (Winoto, Kristianus, and Premachandra 2020) where the authors consider slim networks for deployment on mobile devices.

One of our main results demonstrates that even a single state observation could be sufficient to learn the Game of Life. Similar findings are presented in (Motamedi, Sakharnykh, and Kaldewey 2021), where the authors achieve high performance in an image classification task by removing a significant part of the training data and only keeping the highest quality images. The process of designing or creating training data itself is an actively developing area that has seen significant progress in different areas of machine learning, see e.g. (Ratner et al. 2016; Abufadda and Mansour 2021).

Game of Life as a Neural Network

In the Game of Life, the state of the board changes each turn, depending on the current board configuration, according to

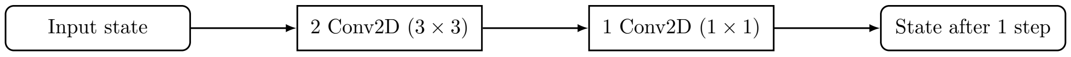


Figure 2: Minimal CNN architecture for the Game of Life network.

the following rules:

1. Any live cell with fewer than two live neighbors dies, as if by underpopulation.
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by overpopulation.
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

Following the established convention, we represent a state of the board in Game of Life by a grayscale image, see Figure 1, where a pixel value of 1 indicates that the cell is alive and a value of 0 indicates that the cell is dead. Thus at a time $t \geq 0$ the state is given as a binary matrix $s_t \in \mathbb{R}^{M\times N}$, where each value $c \in s_t$ represents the condition of a cell at time t with $c = 1$ indicating that the cell is alive and $c = 0$ indicating that the cell is dead. Therefore the Game of Life transition can be viewed as an operator $\mathcal{G} : \mathbb{R}^{M\times N} \to \mathbb{R}^{M\times N}$ acting on the current state s_t and outputting the next state s_{t+1}, i.e.

$$\mathcal{G}(s_t) = s_{t+1}.$$

To establish a neural network representation of the operator $\mathcal{G}$, note that a single step $s_t \to s_{t+1}$ of Game of Life consists of transforming each cell $c \to c'$ as

$$c' = \begin{cases} 1 & \text{if the } 3\times 3 \text{ patch centered at } c \\ & \text{has at least 2 living cells and} \\ & c \text{ has at most 3 living neighbors,} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Therefore in order to make a transition $s_t \to s_{t+1}$, for each cell c one has to know two pieces of information: the condition of the cell c itself and the conditions of the cells in the 3×3 patch centered at cell c. A conventional architecture that can efficiently extract this information with just a few trainable parameters is the *Convolutional Neural Network*, see e.g. (Goodfellow, Bengio, and Courville 2016).

Indeed, the Game of Life operator $\mathcal{G} : \mathbb{R}^{M\times N} \to \mathbb{R}^{M\times N}$ can be represented as a 2-layer convolutional neural network with 23 trainable parameters, see Figure 2. Conceptually, the first layer contains two 3×3 filters, extracting the information on the number of alive neighbors and the condition of the cell itself. This layer can be followed by any reasonable activation function, so we consider ReLU and Tanh — the most popular choices among practitioners. The second layer combines these two features to make a prediction on the cell condition on the next step. Unlike the first layer, the choice of activation function here is more restrictive since the output of $\mathcal{G}$ should only consist of 0 and 1, regardless of the input. As such, we only consider the ReLU activation after the second layer.

Below we provide two minimal configurations with conventional architectures that are capable of capturing the transition rules (1). We note that while it is technically possible to construct a smaller network recreating the rules for Game of Life, it would require either a use of residual connections, unconventional layer structures, or activation functions, which is beyond the scope of the current work, and hence we only consider the fully-CNN architectures.

ReLU Activation. Consider a 2-layer convolutional neural network with the ReLU activation after each layer:

$$s_t \to 2\,\text{Conv2D}(3\times3) \xrightarrow{\text{ReLU}} 1\,\text{Conv2D}(1\times1) \xrightarrow{\text{ReLU}} s_{t+1}$$

The network consists of convolutional filters with the following weights and biases:

$$W_{1,1} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad b_{1,1} = -3$$

$$W_{1,2} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{pmatrix} \qquad b_{1,2} = 3$$

$$W_{2,1} = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \qquad b_{2,1} = 1$$

where $W_{i,j}$ and $b_{i,j}$ are the j-th weights and bias of the i-th layer of the network. Note that for a state $s_t \in \mathbb{R}^{M\times N}$ we get the following output of the first layer:

$$\begin{aligned} \text{ReLU}(W_{1,1}s_t + b_{1,1}) &= [c'_{ij}]_{i=1,j=1}^{M,N} \\ &= \begin{cases} 0 & \text{if } c_{ij} \text{ has} \leq 3 \text{ living neighbors,} \\ > 0 & \text{otherwise.} \end{cases} \end{aligned}$$

$$\text{ReLU}(W_{1,2}s + b_{1,2}) = [c'_{ij}]_{i=1,j=1}^{M,N}$$
$$= \begin{cases} 0 & \text{if a } 3\times3 \text{ patch at } c_{ij} \text{ has} \geq 2 \text{ living cells,} \\ > 0 & \text{otherwise.} \end{cases}$$

Therefore, after the second layer, the value of a cell c' of the output state s_{t+1} is equal 1 if a 3×3 patch centered at a cell c has at least 2 living cells and c has at most 3 living neighbors, or 0 otherwise, which correspond to the Game of Life rules (1).

Tanh Activation. Consider a 2-layer convolutional neural network with the Tanh activation after the first layer:

$$s_t \to 2\,\text{Conv2D}(3\times3) \xrightarrow{\text{Tanh}} 1\,\text{Conv2D}(1\times1) \xrightarrow{\text{ReLU}} s_{t+1}$$

The network consists of convolutional filters with the following weights and biases:

$$W_{1,1} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 3/5 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad b_{1,1} = -2.4$$

$$W_{1,2} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2/5 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad b_{1,2} = -3.6$$

$$W_{2,1} = \begin{pmatrix} 2 \\ -2 \end{pmatrix} \qquad b_{2,1} = -1$$

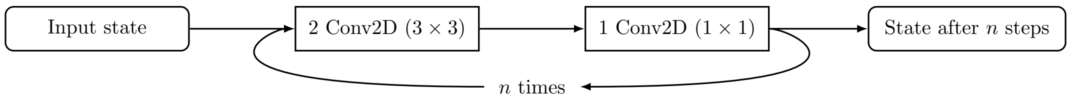


Figure 3: Minimal recursive CNN architecture for the multi-step Game of Life network.

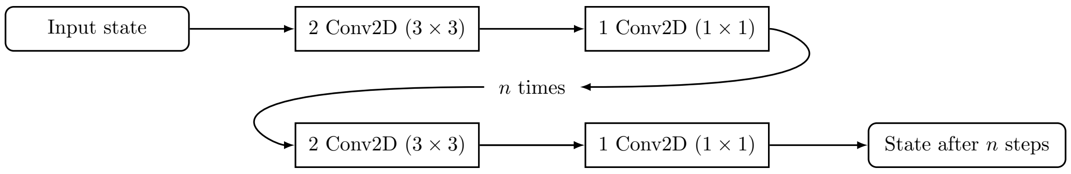


Figure 4: Minimal sequential CNN architecture for the multi-step Game of Life network.

where $W_{i,j}$ and $b_{i,j}$ are the j-th weights and bias of the i-th layer of the network. Then for any cell $c \in \{0;1\}$ with $n \in \{0,1,2,3,4,5,6,7,8\}$ alive neighbors the output of the second layer before the ReLU activation is the following:

	$c=0$	$c=1$
$n=0$	-0.9703	-0.9002
$n=1$	-0.7926	-0.3766
$n=2$	0.0834	1.0621
$n=3$	1.1482	1.0621
$n=4$	0.0834	-0.3766
$n=5$	-0.7926	-0.9002
$n=6$	-0.9703	-0.9862
$n=7$	-0.9960	-0.9981
$n=8$	-0.9995	-0.9997

Thus, after applying the ReLU activation and predicting the condition of a cell c', we obtain a transition corresponding to the Game of Life rules (1). We note that while technically this network does not achieve zero loss, it is capable of reaching 100% accuracy and is easier to train in practice, likely due to a more gradual feature transfer of the Tanh function.

Multi-step Game of Life. In order to evaluate the benefits of a data-centric approach in more challenging scenarios, we consider the task of learning n-steps Game of Life, which becomes increasingly more complex with the increase of the number of steps n. In this formulation, the training data is given in the form of pairs (x, y), where y is the state x after n steps of the Game of Life. In this case the multi-step operator $\mathcal{G}_n$ can be represented in either of the following ways:

- 1-step network, recursively fed into itself n times, which results in a total of 23 trainable parameters, see Figure 3;
- n instances of the 1-step network connected sequentially, which results in a total of $23n$ trainable parameters, see Figure 4.

In our numerical experiments we consider both options. We also note that in this paper we only consider 1- and 2-step formulations of Game of Life learning, since for $n \geq 3$ none of the learning configurations managed to learn the environment regardless of the choices of initialization, optimization algorithm, and hyperparameters. This is due to the constrained nature of our setting as we are only considering networks of minimal architecture.

Training Data Design

A conventional way of obtaining training data for a classification task typically involves generating a sufficiently large number of pairs (x, y) via the methods appropriate for the current scenario (Roh, Heo, and Whang 2019). In the case of n-step Game of Life, x represents the initial state and y represents the same state after n steps. In practice, such a dataset can be obtained by fixing some initial states and computing the corresponding outputs.

A state in Game of Life is represented by a *board* — a 2-dimensional binary matrix where a value of 0/1 represents a dead/alive cell respectively — and a training set is given as a collection of boards. When constructing a training set, one has to decide on the number of training boards, size of each board, and an average board *density* — the percentage of alive cells on the board. Below we explain the choice of these parameters for each of two datasets we use in this paper: the "random" dataset and the "fixed" dataset.

Random Dataset. While there are multiple viable choices for each of these parameters, in the interest of being consistent with the existing literature, we replicate the data collection choices employed in (Springer and Kenyon 2021). Specifically, we implement a data generator to randomly sample boards with an average density of 38%, as such a density minimizes the distribution shift between the inputs and outputs, see Figure 5, and also maximizes the learning success rate, see (Springer and Kenyon 2021, Section 3.4).

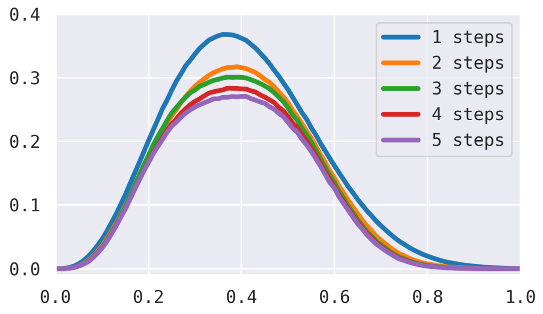


Figure 5: Average board density after multiple steps of the Game of Life.

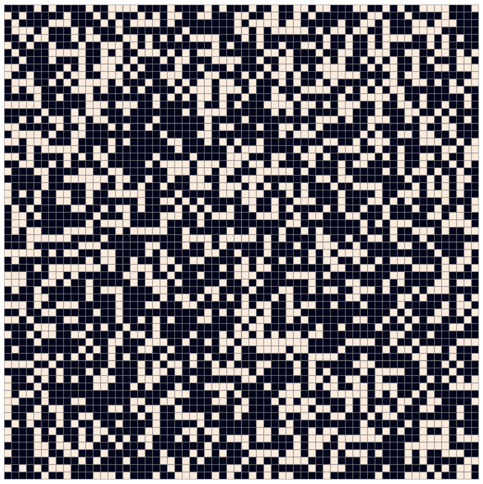


Figure 6: Example of a generated training board.

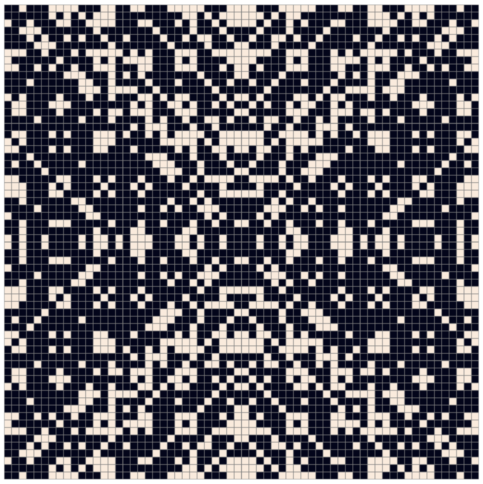


Figure 7: Manually constructed training board.

The size of each training board is set to 64×64 to match the size of our manually designed training board. In order to mitigate the risk of learning on a poorly generated board, we sample a new training board on each iteration of the algorithm when training on the "random" dataset. An example of the generated training board is presented in Figure 6.

Fixed Dataset. A comprehensive answer on the most appropriate training dataset design is highly application-specific and is yet to be found in general. In the context of Game of Life, the main challenges are the redundancy of the patterns in random board that dilute the knowledge and the undecidability of the state dynamics. To address these challenges, we utilize two main concepts in the process of board construction: *symmetry* and *patterns*.

As it follows from the rules (1), in order to make a prediction on the condition of the current state, one does not have to distinguish the neighboring cells; thus, one can deduce that the filters of the first layer should be symmetric about the center. However, it is non-trivial to promote such behavior by utilizing a randomly sampled dataset. To endorse the symmetric structure of the convolutional filters, we make our board symmetric by performing horizontal and vertical reflections. Such an arrangement essentially means that each 3×3 patch is seen by the network 4 times with different orientations, thus promoting symmetry of the filters.

It is evident that the training data must contain a variety of 3×3 patches from which the rules (1) can be learned. While considering every possible 3×3 patch is hardly feasible, as it results in $2^9 = 512$ different configurations, it is also not necessary, as most of them are redundant due to the indistinguishability of the neighbors and the symmetry of the filters. As such, one can prioritize putting the more important patterns in the training board and thereby avoid excessive redundancy, which is the approach we pursue. Some of the patterns we use are presented in Figure 8.

To accommodate the above concepts, we first construct a 32×32 board that contains the critical patterns, and then reflect it horizontally and vertically to harness the power of symmetry, which results in a 64×64 board. Note that the reflection of the board automatically creates additional patterns that are not explicitly included in the original 32×32 board, which allows us to keep the board size relatively small. The constructed board that we use in our numerical examples is presented in Figure 7.

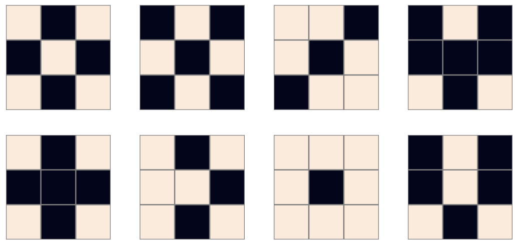
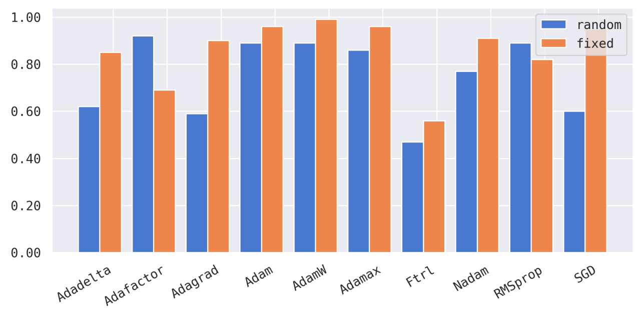


Figure 8: Examples of patterns utilized in construction of the "fixed" training board for the Game of Life.

Numerical Experiments

Our experiments are performed in `Python 3.8` on a personal laptop. Training of all the neural networks is done in `TensorFlow 2.12`. The source code reproducing the presented experiments is provided in Supplementary Materials.

In this section we provide an extensive quantitative analysis to showcase the advantage of a data-centric approach. To this end, we employ all available optimization methods currently implemented in `Tensorflow 2.12`. Specifically, we deploy the following optimization algorithms: `Adadelta` (Zeiler 2012), `Adafactor` (Shazeer and Stern 2018), `Adagrad` (Duchi, Hazan, and Singer 2011), `Adam` and `Adamax` (Kingma and Ba 2014), `AdamW` (Loshchilov and Hutter 2017), `Ftrl` (McMahan et al. 2013), `Nadam` (Dozat 2016), `RMSprop` (Hinton, Srivastava, and Swersky 2012), and `SGD` with momentum (Sutskever et al. 2013). The details for each algorithm and the exact



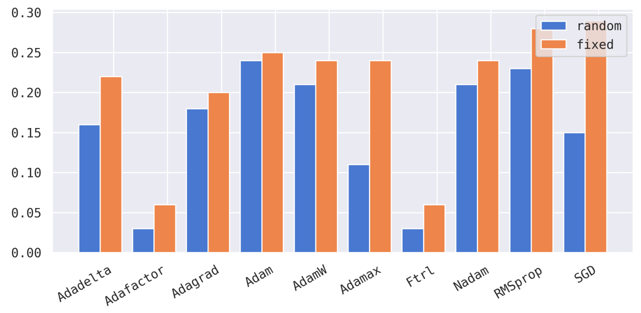


Figure 9: Success rates on 1-step Game of Life for networks with Tanh (left) / ReLU (right) activations.

	Success rate			Number of epochs		
algorithm	random	fixed	change	random	fixed	change
Adadelta	0.61	0.85	+39%	7449	5022	+33%
Adafactor	0.92	0.69	-25%	2403	2590	-8%
Adagrad	0.59	0.90	+53%	2366	1174	+50%
Adam	0.89	0.96	+8%	402	966	-140%
AdamW	0.89	0.99	+11%	313	8992	-2771%
Adamax	0.86	0.96	+12%	294	3078	-946%
Ftrl	0.47	0.56	+19%	6992	1112	+84%
Nadam	0.77	0.91	+18%	2119	1665	+21%
RMSprop	0.89	0.82	-8%	2414	747	+69%
SGD	0.60	0.95	+58%	866	1420	-64%

	Success rate			Number of epochs		
algorithm	random	fixed	change	random	fixed	change
Adadelta	0.16	0.22	+38%	5425	4657	+14%
Adafactor	0.03	0.06	+100%	3530	5127	-45%
Adagrad	0.18	0.20	+11%	655	1176	-79%
Adam	0.24	0.25	+4%	1608	830	+48%
AdamW	0.21	0.24	+14%	4975	714	+86%
Adamax	0.11	0.24	+118%	7384	807	+89%
Ftrl	0.03	0.06	+100%	2835	1380	+51%
Nadam	0.21	0.24	+14%	3869	2310	+40%
RMSprop	0.23	0.28	+22%	633	9718	-1436%
SGD	0.15	0.29	+93%	2470	1570	+36%

Table 1: Results on 1-step Game of Life for networks with Tanh (left) / ReLU (right) activations.

values of the hyperparameters are given in Appendix in Supplementary Materials.

In the presented experiments, for each environment and algorithm we perform 100 learning simulations. Each simulation consists of deploying an algorithm to train a neural network by minimizing the mean square loss on the provided training set — either a sequence of randomly generated boards or a single fixed board. Each training session is run for $10,000$ epoches, which for our environments we found to be sufficient for an algorithm to either converge or plateau. All the algorithms are evaluated on the same test set consisting of 100 randomly generated 100×100 boards and their corresponding states after n steps of Game of Life. A simulation is considered successful if the network's prediction accuracy on the test set achieves 100%.

For each algorithm we measure the percentage of successful simulations and the average number of epoches required to learn the environment. In order to evaluate the advantage of this data-centric approach over the conventional ones, for each experiment we report the relative change in *success* (given by the number of successful simulations) and *efficacy* (given by the average number of epoches required to reach convergence).

1-step Game of Life. The results of learning 1-step Game of Life are presented in Figure 9 and Table 1.

2-step Game of Life with Recursive Networks. The results of learning 2-step Game of Life with recursive (see Figure 3) networks are presented in Figure 10 and Table 2.

2-step Game of Life with Sequential Networks. The results of learning 2-step Game of Life with sequential (see Figure 4) networks are presented in Figure 11 and Table 3.

Discussion. We note that the advantage of designing a fixed training board is not necessarily evident in the 1-step prediction setting, and in some cases results in slower convergence. We hypothesize that this is due to the relative simplicity of the environment, where even the conventional approaches perform well. However, considering the 2-step prediction settings is where one observes the benefits of using custom designed training data, as is apparent from Figures 10, 11 and Tables 2, 3. Overall, our results highlight the importance of a data-centric approach on challenging tasks where conventional approaches might not achieve satisfactory performance.

Conclusion. In this work we study the importance of training data for the efficiency of machine learning algorithms in constrained environments, represented by the multi-step Game of Life prediction tasks. We observe the advantage of a data-centric approach over its conventional counterpart, which becomes especially evident in more challenging environments. Specifically, we demonstrate that training on a single properly-designed data point can be more beneficial than $10,000$ generated data points in terms of both convergence and efficiency. Our results suggest the critical role of the data design process, which supports the need for ML practitioners to work with domain experts in real-life applications.

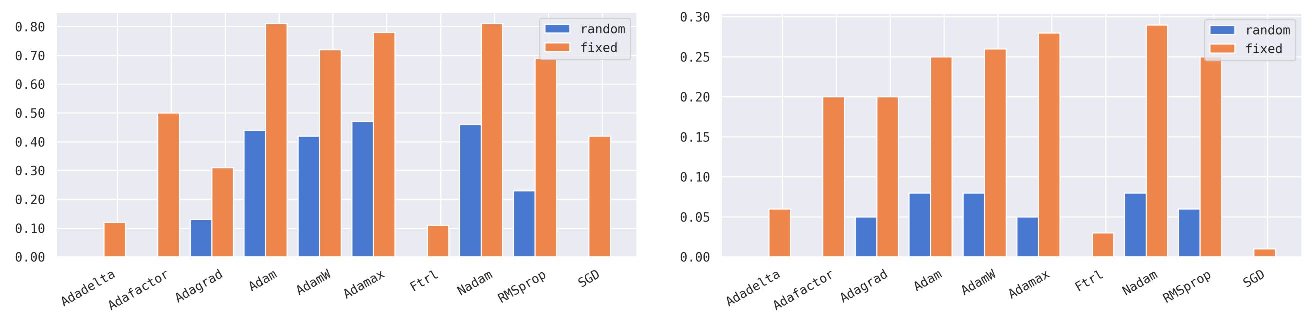


Figure 10: Success rates on 2-step Game of Life with recursive networks with Tanh (left) / ReLU (right) activations.

	Success rate			Number of epochs		
algorithm	random	fixed	change	random	fixed	change
Adadelta	—	0.19	—	—	9598	—
Adafactor	—	0.52	—	—	6839	—
Adagrad	0.14	0.36	+157%	9649	9134	+5%
Adam	0.44	0.81	+84%	7753	4472	+42%
AdamW	0.43	0.72	+67%	9547	3906	+59%
Adamax	0.47	0.78	+66%	3935	5984	-52%
Ftrl	—	0.26	—	—	9794	—
Nadam	0.46	0.82	+78%	8441	7261	+14%
RMSprop	0.26	0.69	+165%	9985	9981	+0%
SGD	—	0.42	—	—	8535	—

	Success rate			Number of epochs		
algorithm	random	fixed	change	random	fixed	change
Adadelta	—	0.08	—	—	7425	—
Adafactor	—	0.21	—	—	9119	—
Adagrad	0.05	0.20	+300%	6023	5917	+2%
Adam	0.08	0.25	+212%	5693	7399	-30%
AdamW	0.08	0.26	+225%	7718	2803	+64%
Adamax	0.05	0.28	+460%	3318	5536	-67%
Ftrl	—	0.03	—	—	7947	—
Nadam	0.08	0.30	+275%	5524	7653	-39%
RMSprop	0.06	0.25	+317%	9845	7151	+27%
SGD	—	0.01	—	—	4349	—

Table 2: Results on 2-step Game of Life with recursive network and Tanh (left) / ReLU (right) activations.

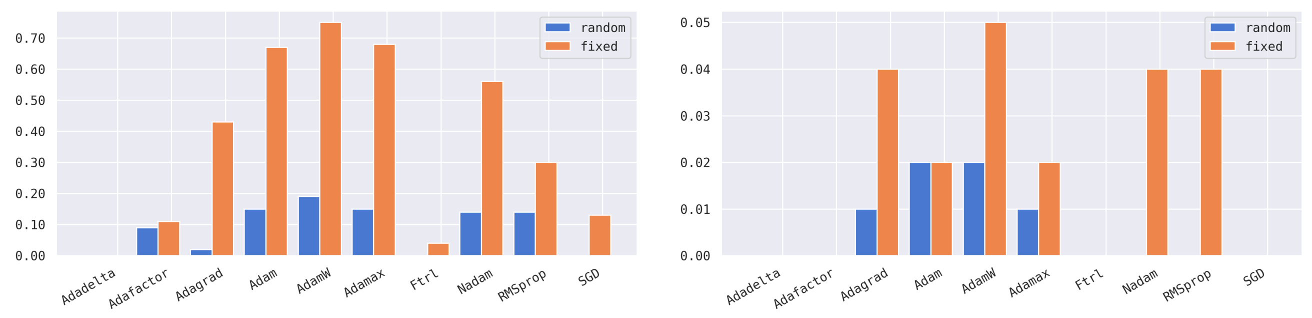


Figure 11: Success rates on 2-step Game of Life with sequential networks with Tanh (left) / ReLU (right) activations.

	Success rate			Number of epochs		
algorithm	random	fixed	change	random	fixed	change
Adadelta	—	—	—	—	—	—
Adafactor	0.08	0.11	+38%	9621	8969	+7%
Adagrad	0.03	0.44	+1367%	9640	6703	+30%
Adam	0.15	0.68	+353%	7994	7122	+11%
AdamW	0.18	0.77	+328%	8076	4113	+49%
Adamax	0.15	0.68	+353%	7151	3536	+51%
Ftrl	—	0.04	—	—	9173	—
Nadam	0.14	0.56	+300%	7976	6811	+15%
RMSprop	0.13	0.31	+138%	9989	9270	+7%
SGD	—	0.13	—	—	9637	—

	Success rate			Number of epochs		
algorithm	random	fixed	change	random	fixed	change
Adadelta	—	—	—	—	—	—
Adafactor	—	—	—	—	—	—
Adagrad	0.01	0.04	+300%	7068	7767	-10%
Adam	0.02	0.02	+0%	5720	5694	+0%
AdamW	0.02	0.05	+150%	8822	6780	+23%
Adamax	0.01	0.02	+100%	6080	2936	+52%
Ftrl	—	—	—	—	—	—
Nadam	—	0.04	—	—	7165	—
RMSprop	—	0.04	—	—	5350	—
SGD	—	—	—	—	—	—

Table 3: Results on 2-step Game of Life with sequential network and Tanh (left) / ReLU (right) activations.

References

Abufadda, M.; and Mansour, K. 2021. A survey of synthetic data generation for machine learning. In *2021 22nd international arab conference on information technology (ACIT)*, 1–7. IEEE.

Adamatzky, A. 2010. *Game of life cellular automata*, volume 1. Springer.

Ahmad, M. A.; Eckert, C.; and Teredesai, A. 2018. Interpretable machine learning in healthcare. In *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics*, 559–560.

Al Janabi, M. A. 2022. Optimization algorithms and investment portfolio analytics with machine learning techniques under time-varying liquidity constraints. *Journal of Modelling in Management*, 17(3): 864–895.

Ammanabrolu, P.; and Hausknecht, M. 2020. Graph constrained reinforcement learning for natural language action spaces. *arXiv preprint arXiv:2001.08837*.

Anik, A. I.; and Bunt, A. 2021. Data-centric explanations: explaining training data of machine learning systems to promote transparency. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–13.

Bae, H.-O.; Ha, S.-Y.; Kang, M.; Lim, H.; Min, C.; and Yoo, J. 2022. A constrained consensus based optimization algorithm and its application to finance. *Applied Mathematics and Computation*, 416: 126726.

Chen, I. Y.; Pierson, E.; Rose, S.; Joshi, S.; Ferryman, K.; and Ghassemi, M. 2021. Ethical machine learning in healthcare. *Annual review of biomedical data science*, 4: 123–144.

Dalal, G.; Dvijotham, K.; Vecerik, M.; Hester, T.; Paduraru, C.; and Tassa, Y. 2018. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*.

Devarajan, H.; Zheng, H.; Kougkas, A.; Sun, X.-H.; and Vishwanath, V. 2021. Dlio: A data-centric benchmark for scientific deep learning applications. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 81–91. IEEE.

Dozat, T. 2016. Incorporating Nesterov momentum into Adam. http://cs229.stanford.edu/proj2015/054_report.pdf.

Dritsas, E.; and Trigka, M. 2022. Data-driven machine-learning methods for diabetes risk prediction. *Sensors*, 22(14): 5304.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Emmert-Streib, F.; and Yli-Harja, O. 2022. What Is a Digital Twin? Experimental Design for a Data-Centric Machine Learning Perspective in Health. *International Journal of Molecular Sciences*, 23(21): 13149.

Gennatas, E. D.; Friedman, J. H.; Ungar, L. H.; Pirracchio, R.; Eaton, E.; Reichmann, L. G.; Interian, Y.; Luna, J. M.; Simone, C. B.; Auerbach, A.; et al. 2020. Expert-augmented machine learning. *Proceedings of the National Academy of Sciences*, 117(9): 4571–4577.

González, D.; Pérez, J.; Milanés, V.; and Nashashibi, F. 2015. A review of motion planning techniques for automated vehicles. *IEEE Transactions on intelligent transportation systems*, 17(4): 1135–1145.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. Convolutional networks. In *Deep learning*, volume 2016, 330–372. MIT press Cambridge, MA, USA.

Gori, M.; Betti, A.; and Melacci, S. 2023. *Machine Learning: A constraint-based approach*. Elsevier.

Grattarola, D.; Livi, L.; and Alippi, C. 2021. Learning graph cellular automata. *Advances in Neural Information Processing Systems*, 34: 20983–20994.

Hinton, G.; Srivastava, N.; and Swersky, K. 2012. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

Hirte, R. 2022. John Horton Conway's Game of Life: An overview and examples.

Horvatha, B.; Gonzalezb, A. M.; and Pakkanenc, M. S. 2023. Harnessing Quantitative Finance by Data-Centric Methods. *Machine Learning and Data Sciences for Financial Markets: A Guide to Contemporary Practices*, 265.

Jo, E. S.; and Gebru, T. 2020. Lessons from archives: Strategies for collecting sociocultural data in machine learning. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, 306–316.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krechetov, M. 2021. Game of life on graphs. *arXiv preprint arXiv:2111.01780*.

Li, Z.; and Dong, J. 2022. Big Geospatial Data and Data-Driven Methods for Urban Dengue Risk Forecasting: A Review. *Remote Sensing*, 14(19): 5052.

Liu, X.-Y.; Xia, Z.; Rui, J.; Gao, J.; Yang, H.; Zhu, M.; Wang, C.; Wang, Z.; and Guo, J. 2022. FinRL-Meta: Market environments and benchmarks for data-driven financial reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 1835–1849.

Loshchilov, I.; and Hutter, F. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Majeed, A.; and Hwang, S. O. 2023. Data-Centric Artificial Intelligence, Preprocessing, and the Quest for Transformative Artificial Intelligence Systems Development. *Computer*, 56(5): 109–115.

McMahan, H. B.; Holt, G.; Sculley, D.; Young, M.; Ebner, D.; Grady, J.; Nie, L.; Phillips, T.; Davydov, E.; Golovin, D.; et al. 2013. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1222–1230.

Miranda, L. J. 2021. Towards data-centric machine learning: a short review. *ljvmiranda921.github.io*.

Motamedi, M.; Sakharnykh, N.; and Kaldewey, T. 2021. A data-centric approach for training deep neural networks with less data. *arXiv preprint arXiv:2110.03613*.

Nguyen, S.; Chan, R.; Cadena, J.; Soper, B.; Kiszka, P.; Womack, L.; Work, M.; Duggan, J. M.; Haller, S. T.; Hanrahan, J. A.; et al. 2021. Budget constrained machine learning for early prediction of adverse outcomes for COVID-19 patients. *Scientific Reports*, 11(1): 19543.

Nye, M.; and Saxe, A. 2018. Are efficient deep representations learnable? *arXiv preprint arXiv:1807.06399*.

Pan, I.; Mason, L. R.; and Matar, O. K. 2022. Data-centric Engineering: integrating simulation, machine learning and statistics. Challenges and opportunities. *Chemical Engineering Science*, 249: 117271.

Perez, G.; Ament, S.; Gomes, C.; and Lallouet, A. 2021. Constrained Machine Learning: The Bagel Framework. *arXiv preprint arXiv:2112.01088*.

Ratner, A. J.; De Sa, C. M.; Wu, S.; Selsam, D.; and Ré, C. 2016. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29.

Rendell, P. 2011. A universal turing machine in conway's game of life. In *2011 International Conference on High Performance Computing & Simulation*, 764–772. IEEE.

Rennard, J.-P. 2002. Implementation of logical functions in the Game of Life. In *Collision-based computing*, 491–512. Springer.

Roh, Y.; Heo, G.; and Whang, S. E. 2019. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, 33(4): 1328–1347.

Seedat, N.; Imrie, F.; and van der Schaar, M. 2022. DC-Check: A Data-Centric AI checklist to guide the development of reliable machine learning systems. *arXiv preprint arXiv:2211.05764*.

Shazeer, N.; and Stern, M. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, 4596–4604. PMLR.

Springer, J. M.; and Kenyon, G. T. 2021. It's hard for neural networks to learn the game of life. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, 1139–1147. PMLR.

Winoto, A. S.; Kristianus, M.; and Premachandra, C. 2020. Small and slim deep convolutional neural network for mobile device. *IEEE Access*, 8: 125210–125222.

Yang, T.-Y.; Hu, M. Y.; Chow, Y.; Ramadge, P. J.; and Narasimhan, K. 2021. Safe reinforcement learning with natural language constraints. *Advances in Neural Information Processing Systems*, 34: 13794–13808.

Yao, J.; Brunskill, E.; Pan, W.; Murphy, S.; and Doshi-Velez, F. 2021. Power constrained bandits. In *Machine Learning for Healthcare Conference*, 209–259. PMLR.

Zahid, A.; Poulsen, J. K.; Sharma, R.; and Wingreen, S. C. 2021. A systematic review of emerging information technologies for sustainable data-centric health-care. *International Journal of Medical Informatics*, 149: 104420.

Zeiler, M. D. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zha, D.; Bhat, Z. P.; Lai, K.-H.; Yang, F.; and Hu, X. 2023. Data-centric AI: Perspectives and Challenges. *arXiv preprint arXiv:2301.04819*.

Zhang, H.; Song, H.; Li, S.; Zhou, M.; and Song, D. 2022. A survey of controllable text generation using transformer-based pre-trained language models. *arXiv preprint arXiv:2201.05337*.

Hyperparameter Search

For each algorithm and environment we perform an extensive hyperparameter search to find the appropriate value of the learning rate λ. Specifically, we perform an exhaustive grid search over the values

$$\lambda \in \{1e{-}1, 3e{-}2, 1e{-}2, 3e{-}3, 1e{-}3, 3e{-}4, 1e{-}4\}$$

by running 100 tests and measuring their corresponding convergence rates. In the case of a tie, the smaller learning rate was used.

We note that, since the random dataset generation is seedable, the hyperparameter search is performed on exactly the same datasets that the algorithms are evaluated on. The results of all tests are provided in Figures 12–17. The value of the learning rate λ providing the highest convergence rate is selected for each algorithm/environment pair, see Tables 4, 5, and 6, where the entry "---" indicates the absence of successful simulations for any of the values of λ.

	Random dataset		Fixed dataset	
algorithm	relu	tanh	relu	tanh
Adadelta	1e-1	1e-1	1e-1	1e-1
Adafactor	3e-2	1e-2	3e-2	3e-2
Adagrad	1e-1	3e-2	1e-1	1e-1
Adam	1e-3	3e-2	3e-3	3e-2
Adamax	3e-4	3e-2	1e-2	1e-3
AdamW	3e-4	1e-2	1e-2	1e-1
Ftrl	1e-1	1e-1	1e-1	1e-1
Nadam	1e-2	1e-3	1e-3	1e-2
RMSprop	3e-3	3e-2	1e-2	1e-2
SGD	3e-2	1e-1	1e-1	1e-1

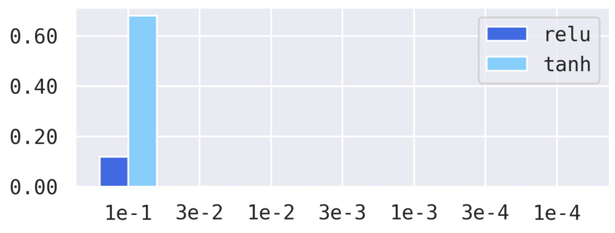
Table 4: Hyperparameters for 1-step Game of Life.

	Random dataset		Fixed dataset	
algorithm	relu	tanh	relu	tanh
Adadelta	---	---	1e-1	1e-1
Adafactor	3e-2	3e-2	3e-2	1e-1
Adagrad	1e-1	1e-1	1e-1	1e-1
Adam	1e-3	3e-3	3e-4	1e-3
Adamax	1e-2	1e-2	1e-3	1e-3
AdamW	3e-3	1e-2	1e-3	1e-3
Ftrl	---	---	1e-1	1e-1
Nadam	1e-3	3e-3	3e-4	1e-3
RMSprop	1e-3	3e-3	1e-3	3e-3
SGD	---	---	1e-1	1e-1

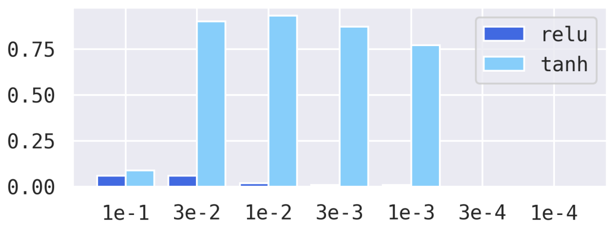
Table 5: Hyperparameters for 2-step Game of Life with recursive network.

	Random dataset		Fixed dataset	
algorithm	relu	tanh	relu	tanh
Adadelta	---	---	---	---
Adafactor	---	1e-1	---	3e-2
Adagrad	1e-1	1e-1	1e-1	1e-1
Adam	1e-3	3e-3	3e-4	3e-3
Adamax	3e-3	3e-2	3e-3	3e-3
AdamW	3e-3	3e-3	1e-3	1e-3
Ftrl	---	---	---	1e-1
Nadam	---	1e-3	1e-3	1e-3
RMSprop	---	3e-3	1e-3	1e-3
SGD	---	---	---	1e-1

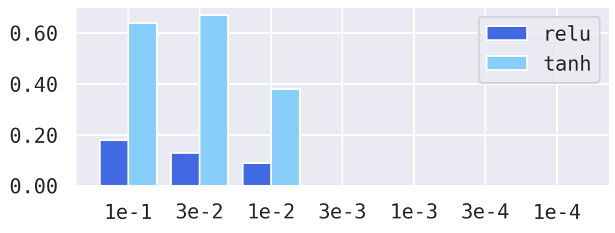
Table 6: Hyperparameters for 2-step Game of Life with sequential network.



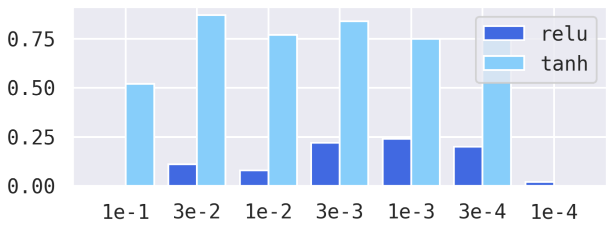
(a) Adadelta



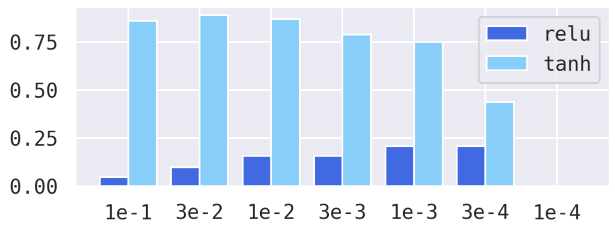
(b) Adafactor



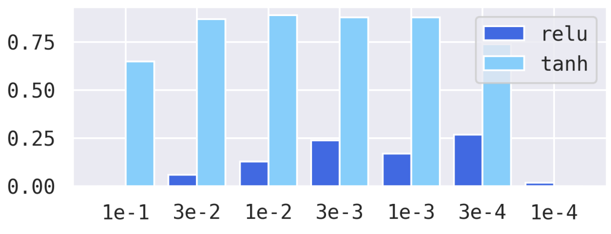
(c) Adagrad



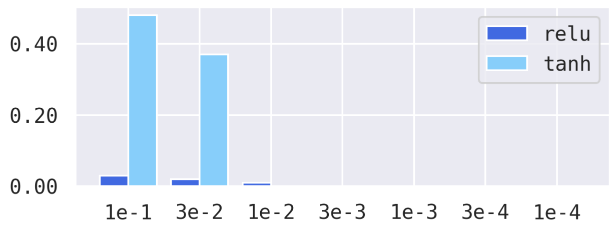
(d) Adam



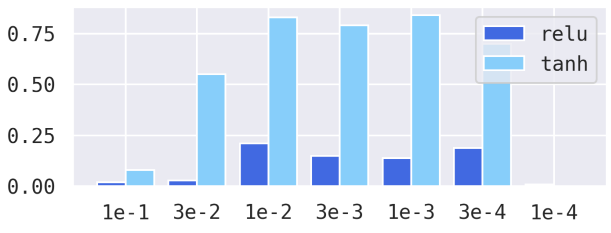
(e) Adamax



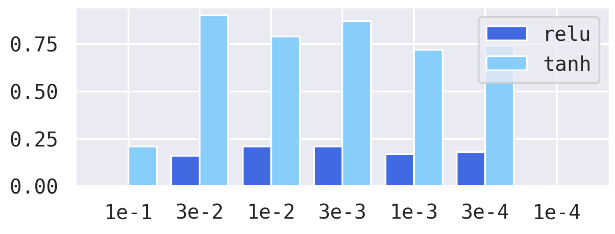
(f) AdamW



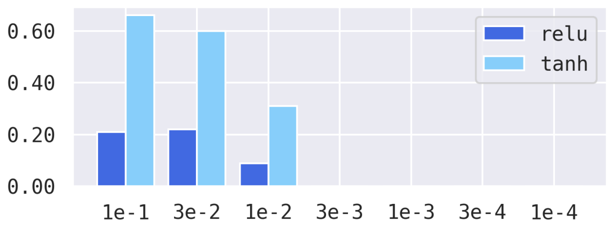
(g) Ftrl



(h) Nadam

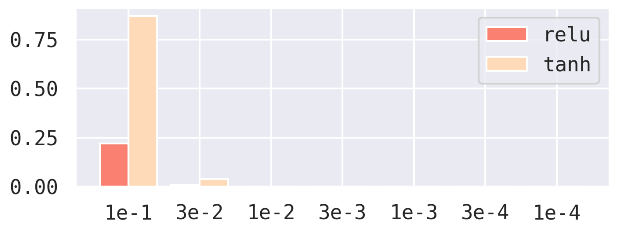


(i) RMSprop

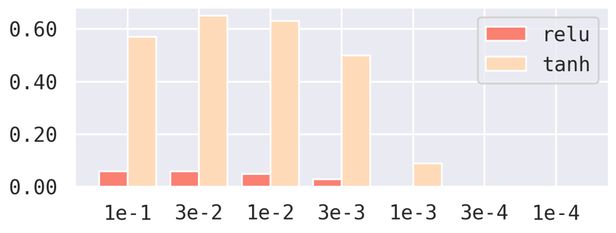


(j) SGD

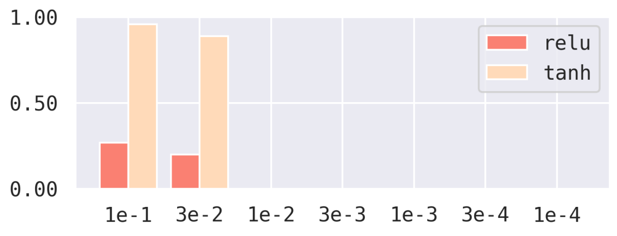
Figure 12: Hyperparameter search for random dataset on 1-step Game of Life.



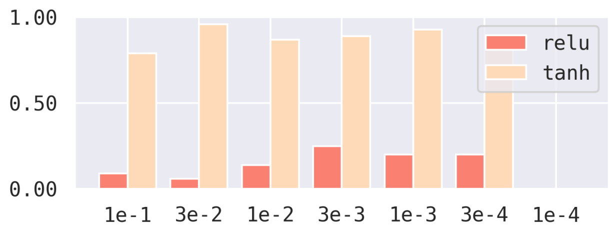
(a) Adadelta



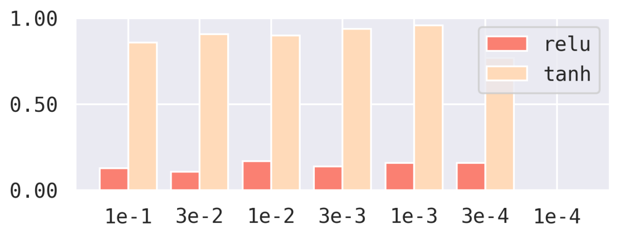
(b) Adafactor



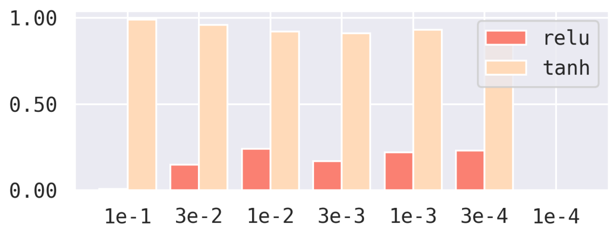
(c) Adagrad



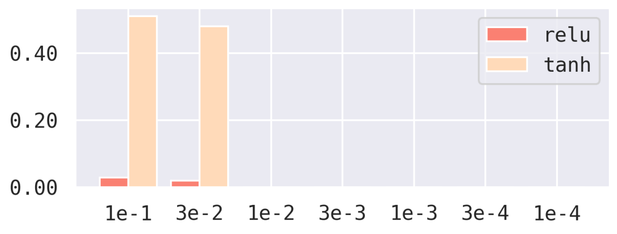
(d) Adam



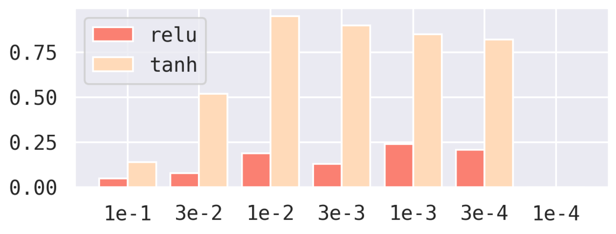
(e) Adamax



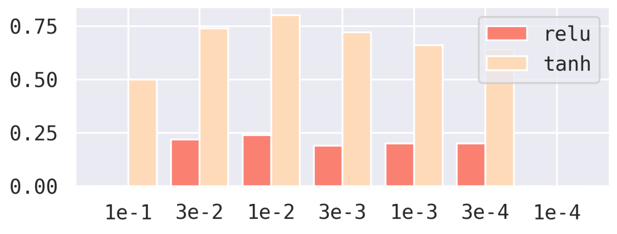
(f) AdamW



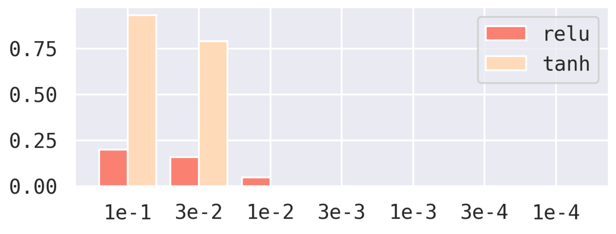
(g) Ftrl



(h) Nadam

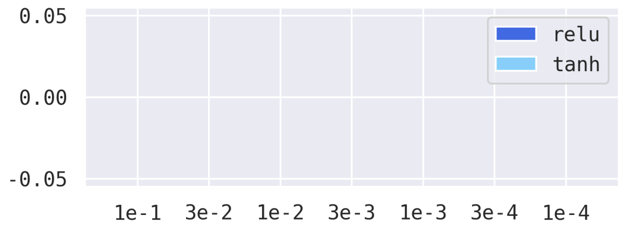


(i) RMSprop

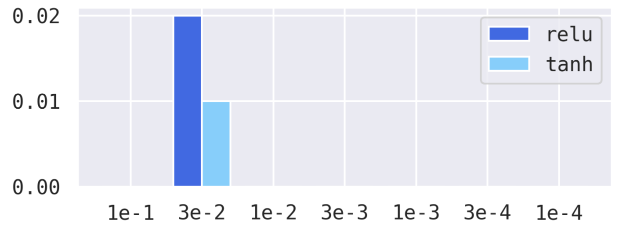


(j) SGD

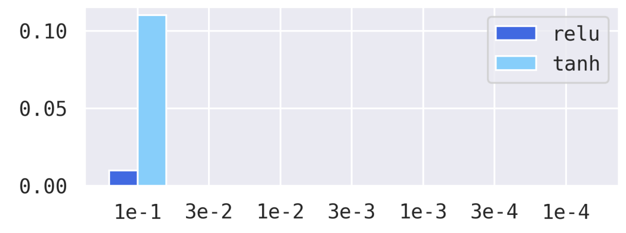
Figure 13: Hyperparameter search for fixed dataset on 1-step Game of Life.



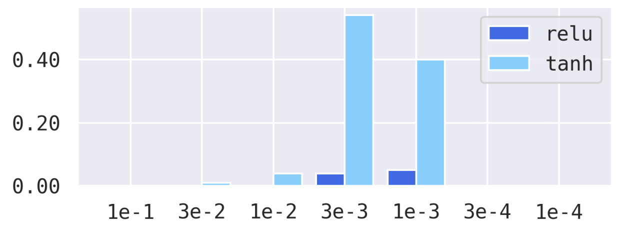
(a) Adadelta



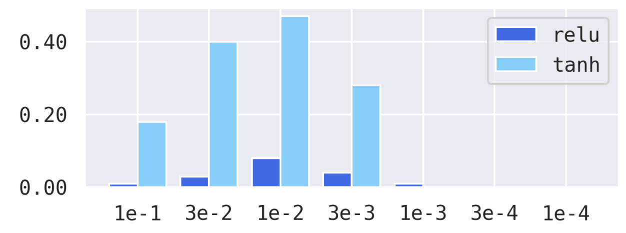
(b) Adafactor



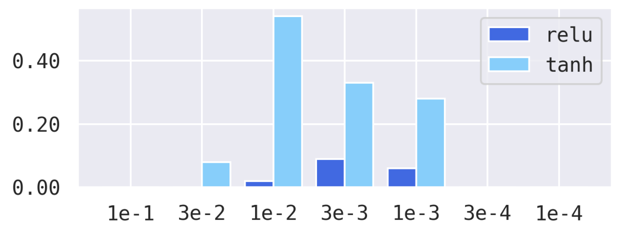
(c) Adagrad



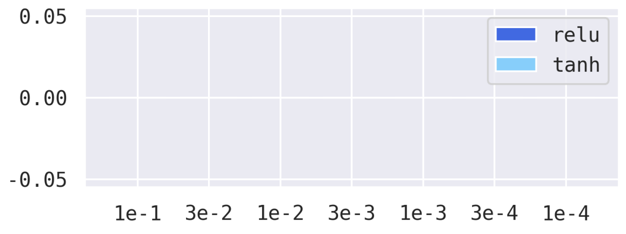
(d) Adam



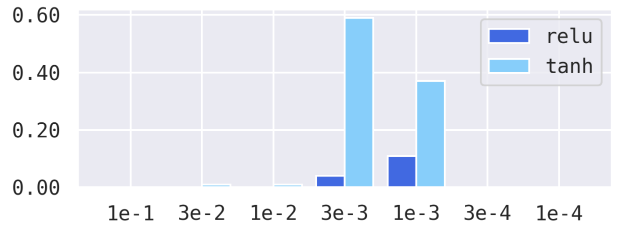
(e) Adamax



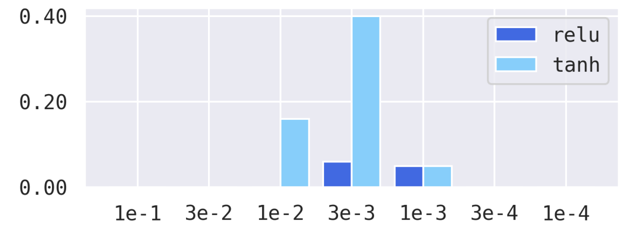
(f) AdamW



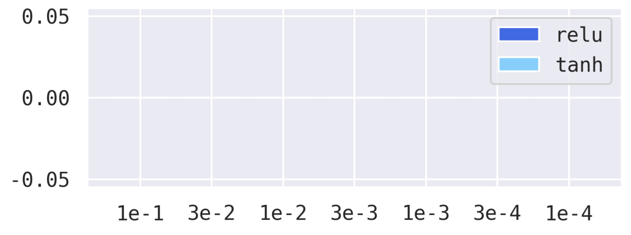
(g) Ftrl



(h) Nadam

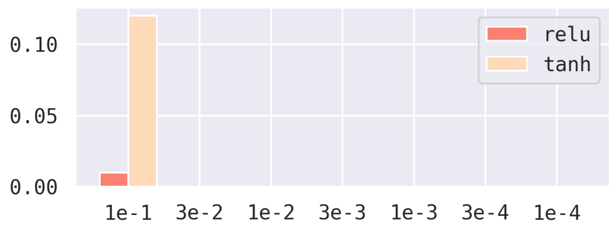


(i) RMSprop

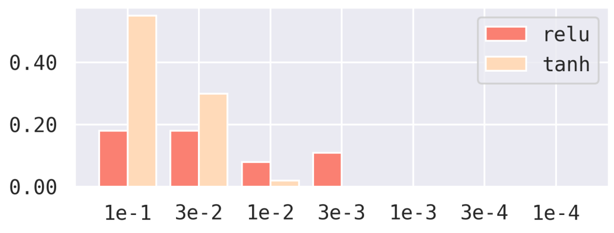


(j) SGD

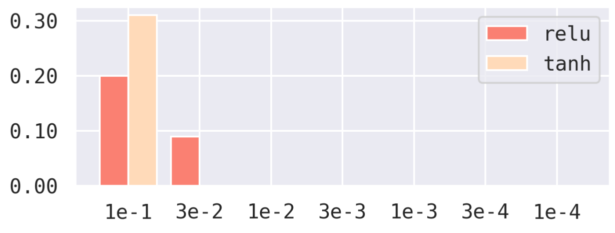
Figure 14: Hyperparameter search for random dataset on 2-step Game of Life with recursive network.



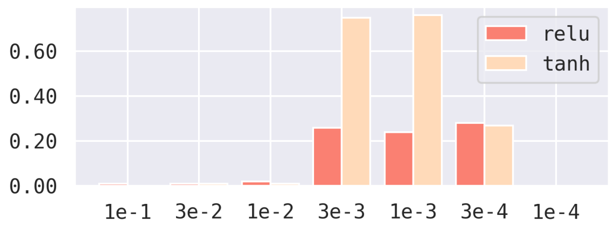
(a) Adadelta



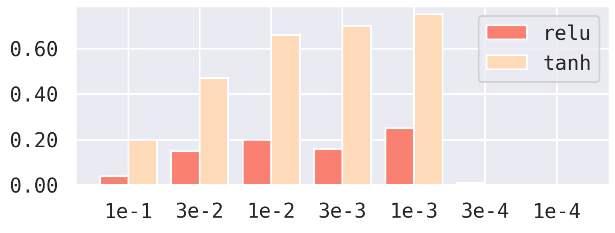
(b) Adafactor



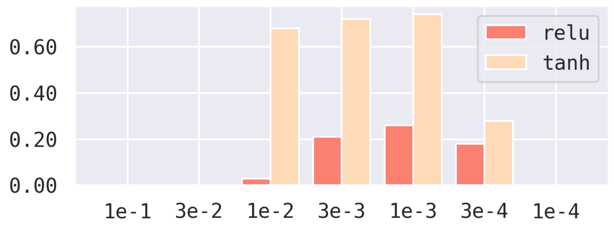
(c) Adagrad



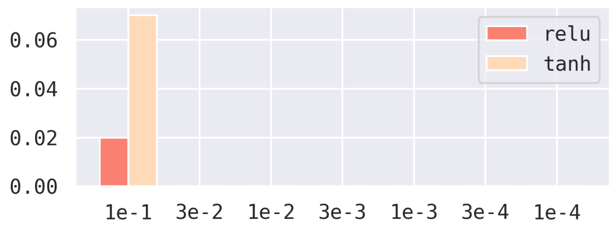
(d) Adam



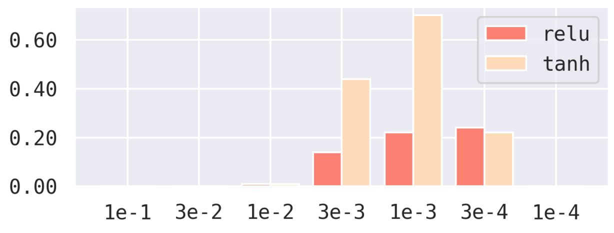
(e) Adamax



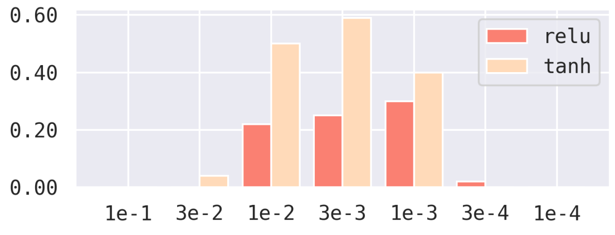
(f) AdamW



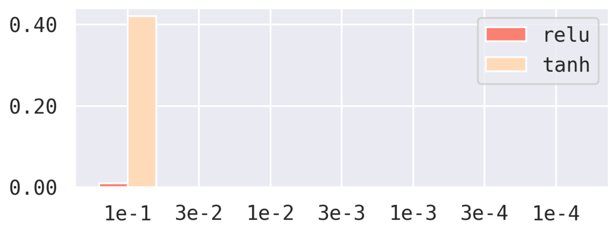
(g) Ftrl



(h) Nadam

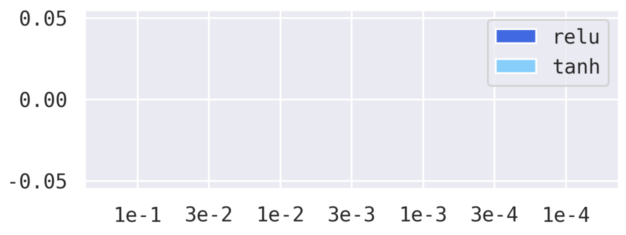


(i) RMSprop

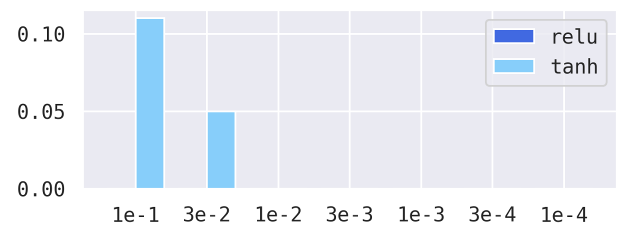


(j) SGD

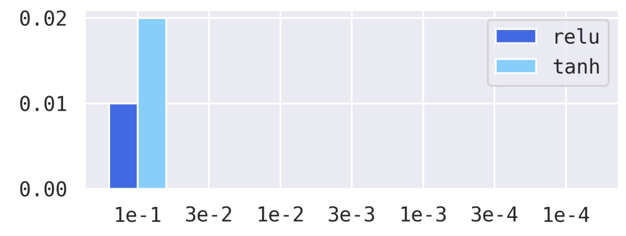
Figure 15: Hyperparameter search for fixed dataset on 2-step Game of Life with recursive network.



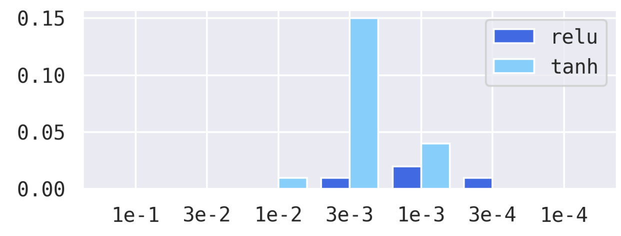
(a) Adadelta



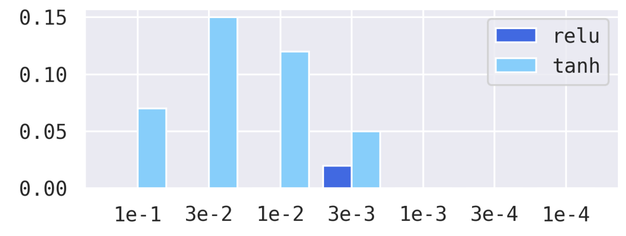
(b) Adafactor



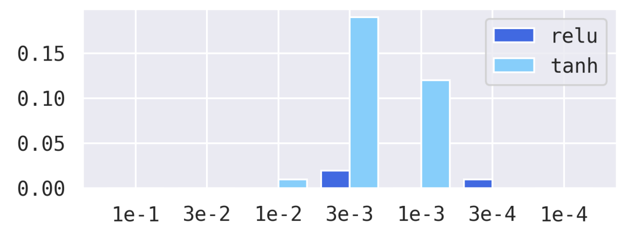
(c) Adagrad



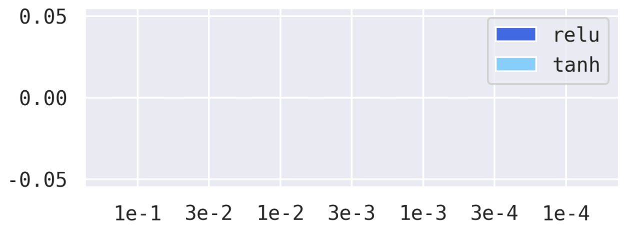
(d) Adam



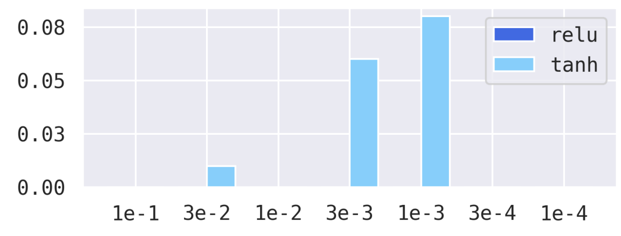
(e) Adamax



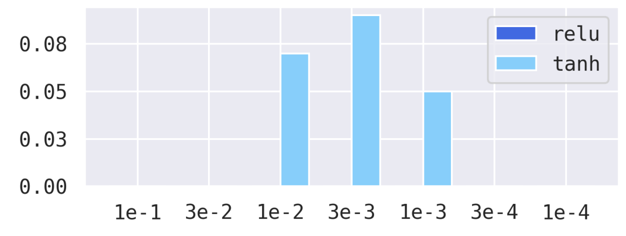
(f) AdamW



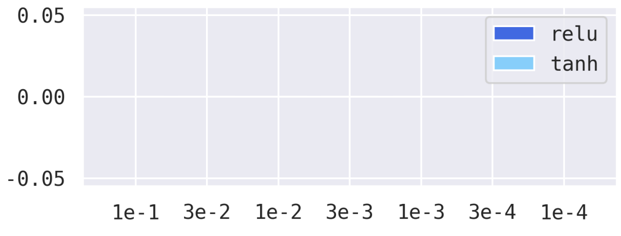
(g) Ftrl



(h) Nadam

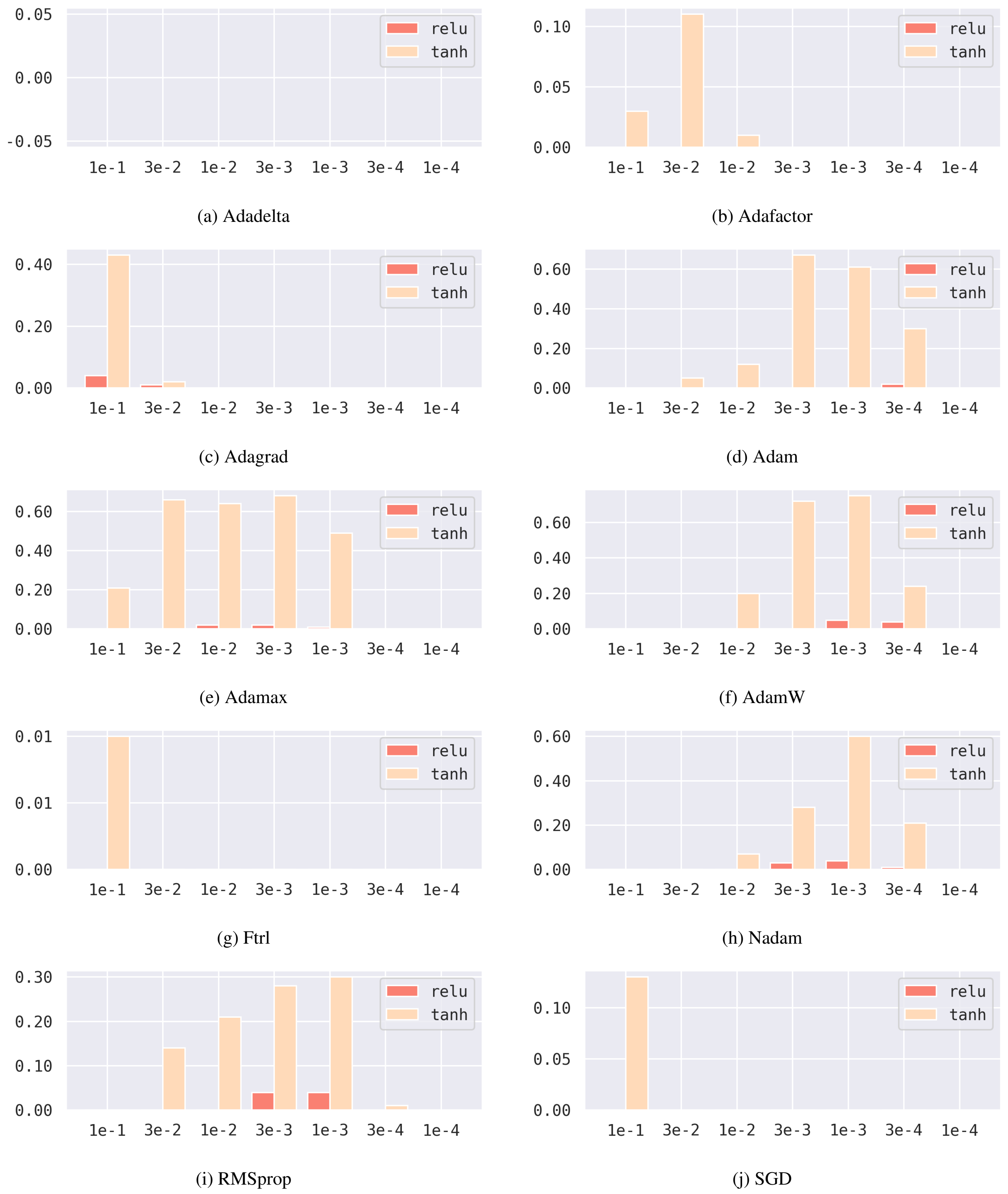


(i) RMSprop



(j) SGD

Figure 16: Hyperparameter search for random dataset on 2-step Game of Life with sequential network.



(a) Adadelta

(b) Adafactor

(c) Adagrad

(d) Adam

(e) Adamax

(f) AdamW

(g) Ftrl

(h) Nadam

(i) RMSprop

(j) SGD

Figure 17: Hyperparameter search for fixed dataset on 2-step Game of Life with sequential network.