Is Grokking a Computational Glass Relaxation?

Xiaotian Zhang[1] **Yue Shang**[1,2] **Entao Yang**[3,*] **Ge Zhang**[1,*]
[1] Department of Physics, City University of Hong Kong, Hong Kong, China
[2] Department of Physics and Astronomy, University of Pennsylvania, Philadelphia, PA, USA
[3] Innovation Campus Delaware, Air Liquide, Newark, DE, USA

Abstract

Understanding neural network's (NN) generalizability remains a central question in deep learning research. The special phenomenon of grokking, where NNs abruptly generalize long after the training performance reaches a near-perfect level, offers a unique window to investigate the underlying mechanisms of NNs' generalizability. Here we propose an interpretation for grokking by framing it as a *computational glass relaxation*: viewing NNs as a physical system where parameters are the degrees of freedom and train loss is the system energy, we find memorization process resembles a rapid cooling of liquid into non-equilibrium glassy state at low temperature and the later generalization is like a slow relaxation towards a more stable configuration. This mapping enables us to sample NNs' Boltzmann entropy (density of states) landscape as a function of training loss and test accuracy. Our experiments in transformers on arithmetic tasks suggests that there is NO entropy barrier in the memorization-to-generalization transition of grokking, challenging previous theory that defines grokking as a first-order phase transition [1]. We identify a *high-entropy advantage under grokking*, an extension of prior work linking entropy to generalizability but much more significant [2]. Inspired by grokking's far-from-equilibrium nature, we develop a toy optimizer, WanD, based on Wang-Landau Molecular Dynamics, which can eliminate grokking without any constraints and find high-norm generalizing solutions. This provides strictly-defined counterexamples to theory attributing grokking solely to weight norm evolution towards the Goldilocks zone [3] and also suggests new potential ways for optimizer design.

1 Introduction

Modern deep learning generally relies on heavily overparameterized neural networks (NNs), which presents surprisingly strong generalization capabilities across diverse tasks [4, 5, 6, 7]. Therefore, understanding the mechanism behind this generalizability is crucial for further advancement in the field. While model generalization performance typically improves with the training performance simultaneously, the grokking phenomenon exhibits dramatically different dynamics, providing an intriguing case for the investigation of generalization. Grokking was first reported in modular arithmetic tasks where researchers used a transformer model [8]. The model can rapidly reach nearly 100% training accuracy but the test accuracy remains very low, suggesting that the model only memorizes the training dataset without actually understanding them. However, after extensive further training, the model exhibits a sudden transition to a state of high generalization where the test accuracy is (almost) at perfect level. This significantly delayed generalization distinguishes grokking from typical learning processes and raises fundamental challenges towards the existing generalization theory [8, 9, 10].

*Corresponding authors: entao.yang@airliquide.com, gzhang37@cityu.edu.hk.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Previous works have shown that the loss landscape of overparameterized NNs possesses infinitely many minima; they exhibit different generalizability and are connected with each other to form a single low-loss manifold [11]. Therefore, many studies have attempted to propose geometric indicators that correlate with the generalizability of minima. One representative hypothesis is that wider minima generalizes better than their narrower analogs [12]. However, it is difficult to measure or even rigorously define the width of a minimum in such a high-dimensional space. While some indicators like sharpness [13] correlates with model generalization empirically, they have also been questioned for the validity in measuring generalizability alone, due to the general scaling invariance in NNs [14]. Other indicators such as the Hessian matrix are too expensive to calculate in deep learning applications [13]. These underscore the limitations of using local geometry alone to predict generalization, which hinders the understanding of complex generalization dynamics like grokking.

Overcoming the limitations of purely geometric descriptions, a very recent work has explored applying concepts from statistical mechanics to NNs, which conceptualizes NN as a physical system where parameters are the degrees of freedom and uses Boltzmann entropy calculated over parameter space to characterize the loss landscape [2]. In this context, entropy measures the *logarithm of the volume* of solutions in the parameter space. The work reveals a *high-entropy advantage* for NNs' generalizability, where the high entropy solutions (the thermodynamic equilibrium state in the context of physics) tend to exhibit better generalization than solutions obtained via classical optimizers like SGD or Adam [2]. This motivates us to employ the entropy framework to investigate the dynamics of grokking. We hypothesize that the transition from memorization (pre-grokking) to generalization (grokking) might involve the network navigating towards the higher-entropy regions. Following the reported protocol [2], we employed the Wang-Landau Molecular Dynamics (WLMD) algorithm to sample transformer neural networks' entropy $S(ln(L_{train}), A_{test})$ as a function of the logarithm of training loss $ln(L_{train})$ and the test accuracy A_{test}, for three different modular arithmetic tasks with varying complexity. We find that the grokking process can be mapped to a computational *glass relaxation*: the training loss of a NN is equivalent to the internal energy of a glass, the memorization training mimics a rapid liquid cooling to form a non-equilibrium glass at low temperature, and the eventual grokking emerges as a slow relaxation process towards a more stable, higher-entropy configuration that generalizes.

The paper is organized as follows: In Section 2, we review the background of generalization, grokking, Boltzmann entropy, and glass physics. In Section 3, we introduce in detail how we sample the entropy landscape of an NN with WLMD. In Section 4, we first present entropy landscapes from three different modular arithmetic tasks, where two of them exhibits grokking while the other one is unlearnable. The results indicate that there is no entropy barrier between the memorization and generalization states. Therefore, grokking is NOT a first-order phase transition, which contrasts with Rubin et al.'s previous work [1]. The grokking NNs also exhibit much more significant high-entropy advantages than those studied in Ref. [2]. We then eliminate grokking by training with constrained weight norm [3] and find that the high-entropy advantage is significantly reduced. These findings inspire us to develop a toy optimizer, WanD, presented in Section 5. We demonstrate that WanD can find generalizing solutions in a competitive efficiency to AdamW while almost eliminating grokking, without any weight-norm constraint or output modifications. Finally, in Section 6 we further discuss related work and situate previous findings on grokking within our proposed framework of computational glass relaxation. We also discuss directions for future research. Code is available at https://github.com/xtzhang28/Grokking.

Our key contributions are as follows:

- **Grokking is NOT a first-order phase transition.** By conceptualizing neural networks as a physical system and sampling its entropy landscape, we find there is no entropy barrier between the pre-grokking memorization state and the final generalizing state, arguing against the first-order phase transition interpretation [1]. Instead, grokking has similar behavior to the formation and relaxation of glass, where the pre-grokking memorization states can be attributed to the fast convergence of train loss minimization, and resembles the rapid cooling of a liquid during glass transition. The subsequent grokking process can therefore be recognized as a slow, computational glass relaxation towards a more stable, generalizing configuration. Such an analogy can describe many reported observations for grokking and inspire potential research directions for NNs, as we detailed later.
- **The high-entropy advantage under grokking is a unique property of generalization.** We find that in the entropy landscape of the grokking NNs, there is a huge gap between

the training curve and the max-entropy curve, which was previously reported as "high-entropy advantage in generalizability" for other NNs and datasets [2]. The high-entropy advantage under grokking, as we found here, is much larger. Furthermore, we show that after eliminating grokking by controlling the weight norm, the high-entropy advantage is much less prominent but still exists. This confirms that the NN without grokking does not have glassy behavior, as they are much closer to the equilibrium state, supporting the correspondence between grokking and glass relaxation. It also reinforces the understanding that higher entropy correlates with better NNs' generalizability [2], broadening its empirical support to our distinct model and task.

- **We design a physics-inspired toy optimizer, WanD, which finds high-norm generalizing solutions without grokking.** WanD is an optimizer based on the Wang-Landau molecular dynamics method. In the modular addition task, WanD has similar time efficiency and generalization performance as AdamW but can eliminate grokking without any explicit regularization (like weight norm constraints). Notably, WanD often finds high-norm generalizing solutions, which challenges a theory attributing grokking solely to weight decay [3]. This also agrees with the conclusion of Kumar et al. [10] that regularization is not always necessary for grokking, but in a strictly-defined situation without output scaling. We argue that grokking is an issue caused by optimization dynamics and can be solved by just improving the optimizer. We believe that WanD points out a potential direction to design new optimizers which has an advantage in finding higher entropy solutions with better generalizability, and our research can encourage further work on applying well-established statistical-physical methodologies for deep learning research.

2 Related Work

Generalization has long been linked to flat minima in the loss landscape [12]. In an over-parameterized NN, different minima form a connected low-loss manifold [11, 15]. Some optimizers like SGD are believed to prefer flatter minima due to its inherent gradient noise [16], explaining why SGD can usually yield better generalization than other optimizers [16]. Therefore, many works focus on developing flatness-based measures and observe correlations with generalizability empirically [13, 17, 18, 19]. However, the usage of flatness as a simple predictor of generalization has been challenged by many works, mainly because it has the dimension of weight [20]. Dinh et al. [14] changed the loss landscape by reconstructing the fitting function without affecting the generalizability, and therefore demonstrated that sharp minima can also generalize. Neyshabur et al. [21] and Feng et al. [20] discussed that flatness itself is not sufficient to ensure generalization, but needs to be combined with the weight norm to obtain a suitable complexity measure. Our results support this because the combination of flatness and weight norm actually reflects the entropy, the logarithm of the parameter space volume, of the solutions with the same generalizability.

Grokking is an anomalous generalization phenomenon that was first discovered by Power et al. [8] in arithmetic tasks, describing a model's sudden transition from random-guess to (nearly) perfect test accuracy long after achieving perfect training performance. Different mechanisms have been proposed to explain this delayed generalization, including: (a) shifts from poor to good representation learning [10, 22, 23, 24]; (b) transitions from dense to sparse predictive subnetworks [25]; (c) slingshot mechanism for adaptive optimizers [26]; (d) first-order phase transition in the network's internal representation [1]; and (e) evolution of weight norms towards the Goldilocks zone [3]. Our analysis from the entropy perspective is consistent with (a)(b)(c), but challenges (d)(e) with well-defined counterexamples. Detailed discussion on connections with existing work in grokking are presented in Section 6.

Boltzmann Entropy defines the number of microstates for a given macrostate in a thermodynamic system. For example, entropy can be defined by the volume in phase space with a certain energy value: $S(U_0) = k_B ln V|_{U=U_0}$, where k_B is Boltzmann constant and V represents the volume in phase space that correspond to the internal energy U_0. Yang et al. [2] applied the concept of Boltzmann entropy to the NNs. This allowed them to improve upon the local perspective of sharpness and weight norm, and established a mapping relationship between generalization and the volume of parameter space. They found that high-entropy NN states usually generalize better than states trained with classical optimizers like SGD, which they define as the high-entropy advantage. This inspired us to use the same approach to study the relationship between entropy and generalization in grokking tasks.

Glass state is a physical state of matter, and is usually formed by rapidly cooling or compressing a liquid. The glass state is not in thermodynamic equilibrium. Therefore, at appropriate temperatures, the glass state will undergo *structural relaxation*- a process where the atomic or molecular arrangement in a non-equilibrium glass gradually evolves toward a more stable, more equilibrated configuration over time. This relaxation process is generally regarded as a kinetic process [27]. Winter and Janssen [28] compared deep NNs with structural glass and found many similarities, including the relaxation dynamics at low temperatures. As we will show in this paper, grokking is thermodynamically similar to glass relaxation, so a promising future direction is to explore whether the conclusions in [28] holds in our setup.

3 Methodology

In this section, we will first define the modular arithmetic task, then introduce the concept of entropy into NNs, and finally present our sampling method: the Wang-Landau Molecular Dynamics algorithm. Code available at https://github.com/xtzhang28/Grokking.

3.1 Problem definition

The training part of our experiment is same as the setup of Nanda et al. [29], that is, a one-layer transformer model trained in modular arithmetic tasks. The data consists of two parts: prompts $\langle x\rangle\langle y\rangle\langle p\rangle$ and answer $\langle x \circ y\rangle$. In the three tasks we study, the answers are determined by the following equations:

$$\begin{aligned} x \circ y &= x + y \ mod \ p \text{ for } 0 \leq x, y < p \\ x \circ y &= x^2 + y \ mod \ p \text{ for } 0 \leq x, y < p \\ x \circ y &= x^3 + xy^2 + y \ mod \ p \text{ for } 0 \leq x, y < p \end{aligned} \tag{1}$$

Here, x, y, and p are all natural numbers. In this paper, p is set to a prime number 67. The entire dataset will be divided into a training set and a test set with a 50% fraction, by a fixed random seed. In the Appendix Figure 4, we provide a visual image of the full dataset. Transformer hyperparameters and more training details are also given in the Appendix A.

3.2 Entropy in neural networks

There are many ways to define entropy in physics and information theory. In this work, we apply the definition of Boltzmann entropy $S = k_B \ln V$ to NNs. Here, k_B is the Boltzmann constant, which we set to 1 for simplicity. In physics, V is the volume of the phase space that corresponds to a particular set of observables (energy, magnetization, etc.). For an NN, we define V as the volume of the parameter space that corresponds to a particular training loss and test accuracy, and therefore S is also a function of these two quantities. To ensure that V is finite, we limit the parameter space by setting different reflective bounds for the parameters of different layers. These bounds are designed to cover the range of parameters typically reached during normal training while avoiding being excessively large. In Sec. 4.2, we avoid applying these reflective bounds to study the entropy advantage when grokking is suppressed. Instead, we limit our study to a hyperspherical shell of weight norm 30, a method known to effectively prevent grokking [3]. Considering grokking often occurs when the training loss is small, our results are presented in terms of its logarithm, $S(ln(L_{train}), A_{test})$. Here, L_{train} denotes the training cross-entropy loss, and A_{test} represents the smoothed test accuracy.

The entropy landscape we calculate cannot reach infinite resolution. For computational purposes, we divide $S(ln(L_{train}), A_{test})$, which is a 2D function, into rectangular bins, where each bin corresponds to the entropy of the NN parameters within a certain range of training loss and test accuracy. The Wang-Landau Molecular Dynamics method allows us to calculate $S(ln(L_{train}), A_{test})$ with an arbitrary zero point. In other words, the absolute value of the entropy is meaningless, but the entropy difference between any two bins is meaningful. Therefore, we can subtract a constant from the entropy in each bin so that the minima-bin entropy is always 0.

3.3 Wang-Landau Molecular Dynamics (WLMD)

In this section, we will introduce the details of WLMD algorithm and its implementation in NNs. The key advantage WLMD is its ability to avoid becoming trapped [30]. In contrast to standard samplers, which tend to oversample the high-entropy regions, WLMD achieves uniform sampling across all regions of the entropy diagram upon convergence [31]. This is accomplished by maintaining a running estimate of the entropy landscape, which introduces an 'entropy gradient' that drives the system away from previously visited states. The convergence of the WLMD is established in [32]. For a physical system made of atoms or particles of other length scales, one can use WLMD to calculate the entropy as a function of the potential energy $S(U)$. This is done by simulating particles with the following equation of motion [30]:

$$\frac{d^2 q_i}{dt^2} = -\frac{T_0}{m_i}\frac{\partial S(U,t)}{\partial U}\frac{\partial U(q_1, q_2, ..., q_n)}{\partial q_i}, \tag{2}$$

where $q_1, q_2, ..., q_i$ is the coordinate of each particle, T_0 is the simulation temperature, m_i is the mass of the particle, t is the simulation time, $S(U,t)$ is an entropy function that we update over time, and U is the potential energy of the system. Initially, $S(U, t=0) = 0$. During the simulation, we repeatedly update $S(U,t)$:

$$S(U, t+\Delta t) = S(U,t) + f_{WL}(t)\delta(U - U_0), \text{ where } f_{WL}(t) = F_{WL} \min\{t/t_{WL}, t_{WL}/t\}. \tag{3}$$

Here the Wang-Landau factor $f_{WL}(t)$ quantifies how fast we update the entropy function. In order to ensure the convergence of the WLMD algorithm, usually we make it increase linearly with time to a maximum value F_{WL} and then decay inversely. δ is a Dirac function, and we approximated it with a narrow Gaussian function in the simulation. U_0 is the potential energy of the system at this moment.

For NNs, we can set the temperature T_0 and the mass m_i to 1, which does not affect the correctness of the WLMD algorithm. The particle coordinates $q_1, q_2, ..., q_n$ are replaced by the NN parameters $c_1, c_2, ..., c_n$, and the entropy we considered is $S(x_{WL}, y_{WL})$. We choose $ln(L_{train})$ as the energy in a broad sense x_{WL}, and A_{test} as the order parameter y_{WL}. The parameter motion follows

$$\begin{aligned}\frac{d^2 c_i}{dt^2} = -\frac{\partial S(x_{WL}, y_{WL}, t)}{\partial c_i} = &-\frac{\partial S(ln(L_{train}), A_{test}, t)}{\partial ln(L_{train})}\frac{\partial ln(L_{train}(c_1, c_2, ..., c_n))}{\partial c_i} \\ &-\frac{\partial S ln(L_{train}), A_{test}, t)}{\partial A_{test}}\frac{\partial A_{test}(c_1, c_2, ..., c_n)}{\partial c_i}.\end{aligned} \tag{4}$$

The term $\frac{\partial S}{\partial ln(L_{train})}$ and $\frac{\partial S}{\partial A_{test}}$ can be computed via numerical differentiation between adjacent bins, while $\frac{\partial ln(L_{train})}{\partial c_i}$ is the gradient of $ln(train\ loss)$ over parameters. Since A_{test} is discrete, we approximate its gradient using the sigmoid function, following the approach in [2]. Further details are also provided in Appendix A.3. Reflective boundary conditions are used to implement parameter constraints for results in Sec. 4.1, and for results in Section 4.2 we restricted the parameter weight norms directly by weight rescaling (both detailed in Sec. 3.2).

The initialized velocity of the parameters follows the Maxwell–Boltzmann distribution, which is a random Gaussian distribution with mean 0 and variance $k_B T/m = 1$. In order to maintain constant temperature during the simulation, that is, to conserve the kinetic energy of the parameters, we add the Langevin kinetic term to the velocities:

$$\frac{dc_i}{dt}(t+\Delta t) = \frac{dc_i}{dt}(t) + \frac{d^2 c_i}{dt^2}\Delta t - c_f \frac{dc_i}{dt}(t) + \sqrt{c_f(2-c_f)k_B T}v_{rand} \tag{5}$$

Here, the definition of $\frac{d^2 c_i}{dt^2}$ is given in Equation 4, c_f is the friction coefficient, and v_{rand} is the random velocity follows a standard Gaussian distribution. In the NN parameters updating process, by continuously adding entropy to the entropy landscape, $S(x_{WL}, y_{WL}, t)$ will eventually converge to the true entropy distribution $S(x_{WL}, y_{WL})$.

Due to space limitations, more details of WLMD and the complete algorithm are described in the Appendix A.3. To the best of our knowledge, WLMD is currently the most efficient entropy sampling algorithm, as existing alternatives are limited to models with only hundreds of parameters [2]. The computational complexity of WLMD scales linearly with the number of NN parameters, and the entropy landscape sampling can be parallelized across multiple GPUs. This enables entropy sampling for large-scale models, such as transformers with hundreds of thousands of parameters.

4 Grokking: Glass Relaxation Process in Neural Network

4.1 No entropy barrier in entropy landscape

In thermodynamics, a phase transition is marked by a significant change in an order parameter—a macroscopic, measurable quantity like density during a liquid–gas transition. By analogy, the learning dynamics of NNs exhibiting grokking appears to mimic the characteristics of a phase transition: starting with a rapid progress from underfitting to memorization (low test accuracy), the NN undergoes an abrupt qualitative shift to generalization (high test accuracy) after a prolonged training period. Given this, we identify the test accuracy as the order parameter for characterizing grokking. Just as a physical order parameter changes distinctly between material states, the definitive increase in test accuracy can track the NN's transition into the generalization regime, making it a natural and compelling macroscopic indicator of this critical learning phenomenon.

A previous study suggests that grokking is a first-order phase transition process from memorization to generalization [1]. They used the Langevin dynamic optimizer to train two two-layer nonlinear teacher-student models and found that the memorization and generalization states of the NN were located at two free-energy minima, thus believing that grokking is a first-order phase transition that requires crossing a free-energy barrier. We directly examined this statement by computing the entropy landscape as a function of training loss and test accuracy to search for such a barrier. We examine three algorithmic tasks on a one-layer transformer model. The three tasks range from easy to learn, to difficult to learn, and to impossible to learn. The training results are also given in the Appendix Figure 5, where grokking is observed for the first two tasks. Using the WLMD method, we sample the entropy landscape of training loss and test accuracy as presented in Figure 1. The three entropy landscapes are the results after 60M, 80M, and 20M WLMD update epochs respectively, where these durations were chosen as the points where the entropy map exhibited no substantial further change. We ran 8 WLMD processes on 4 GeForce RTX 4090 GPUs, requiring approximately 100 GPU hours per 1M epochs. Additional computational details are provided in Appendix A.2.

For the first two tasks that exhibit grokking, the training trajectory is roughly "L" shaped. In the vertical part of the trajectory, i.e., from the memorization state (lower left part) to the generalization state (upper left part), we observed no entropy barrier. Since there is neither an entropy barrier nor an energy (train loss) barrier, our results disproves the view that grokking requires crossing a free-energy barrier. In addition, we find that the equilibrium (maximum entropy for a given training loss) curve in the low train loss region is continuous, which also denies the first-order phase transition, as such transitions require discontinuity of the order parameter (test accuracy). For the third unlearnable task, the equilibrium state also fails to generalize, although some states present a higher test accuracy (about 20%) than random selection (about 1.5%).

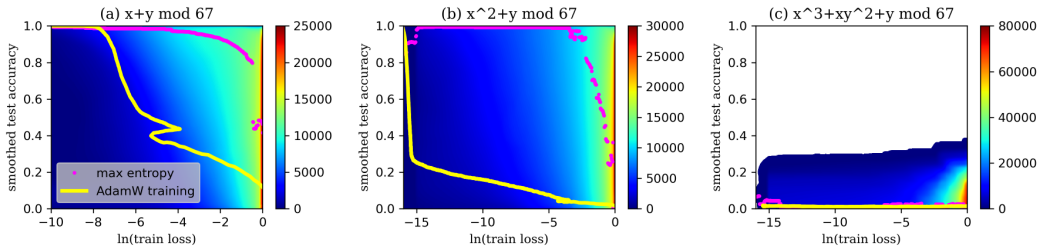

Figure 1: Entropy landscape of three different arithmetic tasks : (a) $x + y \ mod \ 67$ is easy to learn, max entropy state can generalize better than training line. (b) $x^2 + y \ mod \ 67$ is hard to learn, the generalization advantage of the max entropy state over the training results under the same training loss is very significant. (c) $x^3 + xy^2 + y \ mod \ 67$ cannot be learned by training, max entropy state also can not generalize.

We also found a huge gap between the AdamW training curve and the equilibrium state curve. We define this significant advantage of the equilibrium state over the training curve as the high-entropy advantage under grokking. This result shows that grokking is actually a kinetic process rather than a phase transition in the thermodynamic sense: The NN is far from the equilibrium state during the

initial part of the training process, and finally reaches the equilibrium state, which generalizes well, after a long time. This process is similar to glass formation and relaxation: When the liquid is cooled rapidly, the molecules do not have enough time to rearrange themselves into a crystal or crystal-like structure, forming a disordered glassy state (non-equilibrium state), which then gradually evolves toward a better equilibrated state through slow structural relaxation.

4.2 Neural network without grokking does not exhibit glassy behavior

The analogy between grokking and glass relaxation can be further verified by examining the entropy landscape after eliminating grokking. According to our conjecture, just as a slowly cooled liquid will maintain thermodynamic equilibrium and will not enter the non-equilibrium glassy state, an NN without grokking will be trained along the equilibrium curve as the training loss decreases, steadily reaching the generalization state. The study by Liu et al. [3] suggested that grokking can be eliminated by limiting the weight norm of the NN parameters in an appropriate range. Following this insight, we implemented a strict weight norm constraint in our study: after each iteration step, we compute the weight norm of the neural network parameters and scale them proportionally to maintain a fixed norm of 30. The training results, presented in Appendix Figure 6, show that grokking can be greatly eliminated under this condition. We examine three transformer models with varying widths (model dimensions). For each entropy sampling task, we use 4 NVIDIA H100 GPUs to run 8 processes, totaling 60M epochs and consume nearly 4,000 GPU hours. Further computational details are available in Appendix A.2.

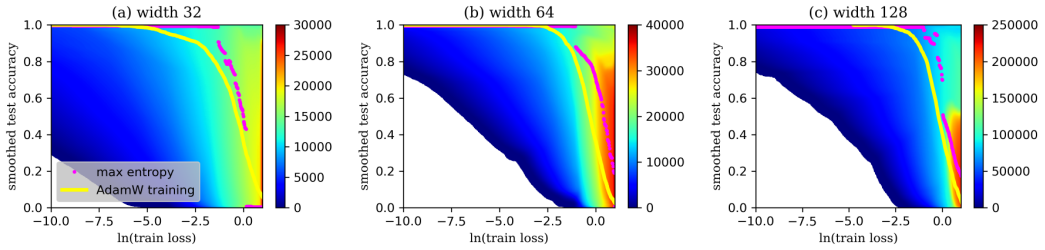

Figure 2: Entropy landscape of modular addition ($p = 67$) task, sampling in three transformer models that have different widths. Here weight norm is constrained to a fixed value of 30. Compared with the grokking NNs, the high-entropy advantage is significantly weakened but still exists.

Figure 2 shows that after controlling the weight norm at 30, the AdamW training curve in the entropy landscape is very close but still slightly below the equilibrium state. Moreover, under this restriction the entropy landscape cannot explore severe memorization states, which is the computational glassy state in our analogy. This result further illustrates the correspondence between glass and grokking. In addition, we note that the high-entropy advantage still exists even in the network where grokking is eliminated. This agrees with the work of Yang et al. [2] that high-entropy advantages are prevalent in various deep learning tasks and there is a robust connection between equilibrium and generalization.

5 WanD: Eliminating grokking without regularization

The above results show that the equilibrium state at low training loss has high generalizability regardless of whether the weight norm of the network is restricted. Therefore, we believe that grokking is a dynamic process caused by the optimizer's inefficient exploration of the loss landscape basins. The WLMD method we used is good at sampling high entropy states of NNs, which inspires us to design an optimizer by imitating WLMD, that can find high entropy states and therefore should also eliminate grokking. Here, we design an optimizer, WanD, based on the one-dimensional Wang-Landau Molecular Dynamics. The continuous-time dynamics of WanD is as follows.

$$\frac{d^2\theta(t)}{dt^2} = -\frac{\partial S(ln(L_{train}(\theta(t))))}{\partial ln(L_{train}(\theta(t)))} \cdot \frac{\partial ln(L_{train}(\theta(t)))}{\partial \theta(t)} + \eta(t) - \lambda \frac{d\theta(t)}{dt}, \tag{6}$$

where $\theta(t)$ is a tensor of the NN parameters in time t, and $ln(L_{train})$ is the logarithm of the cross-entropy training loss. $\eta(t)$ is a noise given by $\langle \eta_i(t)\eta_j(t')\rangle = \lambda(2-\lambda)k_BT\delta_{ij}\delta(t-t')$, $k_BT = 1$

is a temperature term, and λ is a friction coefficient. When sampling the entropy landscape using WLMD, we obtain the corresponding training loss and test accuracy of the NN to locate where to add entropy. For the WanD optimizer, its training process samples the entropy landscape $S(ln(L_{train}))$ based solely on training loss, due to the fundamental constraint that it can only access training data. In addition, we set up an appropriate entropy bias to accelerate the NN's exploration at low training loss area. The overall WanD training procedure is summarized in Algorithm 1.

Algorithm 1 WanD Optimizer

Require: Transformer model $T(\cdot)$, the entropy bias S_{bias}, training data $(X_{train}, y_{train})$and test data(X_{test}, y_{test})

Ensure: Transformer model T reach generalization: $Accuracy(T(X_{test}), y_{test}) \approx 100\%$

The initial entropy: $S \leftarrow S_{bias}$

for each optimize epoch **do**

Compute the logarithm of the train loss:
$ln(L_{train}) = ln(CrossEntropyLoss(T(X_{train}), y_{train}))$;

Compute entropy gradient $x_{grad} = \frac{dS(x_{WL})}{dx_{WL}}$

Compute $\frac{d^2c_i}{dt^2} = -\frac{\partial ln(L_{train})}{\partial c_i} x_{grad}$

for each parameter c_i **do**

Update the parameter velocity: $\frac{dc_i}{dt} \leftarrow \frac{dc_i}{dt}(1 - c_f) + \frac{d^2c_i}{dt^2}\Delta t + v_{Langevin}$

Update the parameter: $c_i \leftarrow c_i + \frac{dc_i}{dt}\Delta t$

end for

Update the Wang-Landau factor by Equation 3: $f_{WL} \leftarrow F_{WL} min\{t/t_{WL}, t_{WL}/t\}$

Update the entropy landscape by Equation 3: $S(x_{WL}) \leftarrow S(x_{WL}) + f_{WL}\delta(x - x_{WL})$

Verification:$A_{test} = Accuracy(T(X_{test}), y_{test})$

end for

The training results for modular addition task using WanD or AdamW are shown in Figure 3. Both methods have comparable performance, achieving generalization after $\sim 2,000$ optimization steps. WanD has a shorter running time, taking about 50 seconds per 1K steps compared to roughly 72 seconds for AdamW on a single GeForce RTX 4090 GPU. During the training process of WanD, the weight norm of the NN parameters continued to increase, eventually reaching about 175, indicating that the NN can generalize at high weight norm. This agrees with the work of Kumar et al. [10] that regularization is not necessary for grokking, but in a more strictly-defined context as we do not scale NN's output, which could be taken as effectively changing the weight norm. We present additional training results for WanD and AdamW in the Appendix B.3 on the same task but eliminating grokking by increasing the training dataset split fraction.

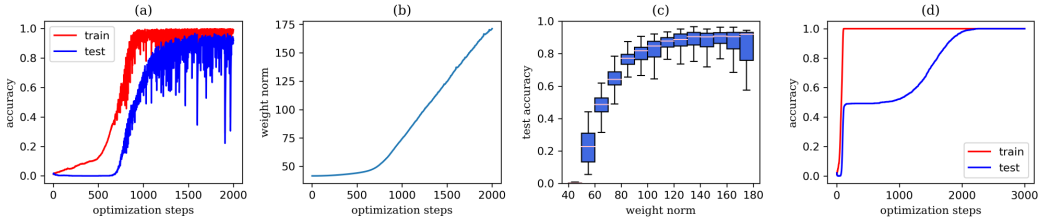

Figure 3: Using (a)(b)(c) WanD and (d) AdamW to train a one layer Transformer on modular addition ($p = 67$) task. (a) Train and test accuracy over time. The grokking is largely eliminated while training with WanD. (b) Weight norm over time. (c) Test accuracy over weight norm. These (a, b, c) together demonstrate that the model can generalize at high weight norm. (d) Train and test accuracy over time. Both WanD and AdamW use the same learning rate.

6 Conclusions and Discussion

6.1 Conclusions

In this work, we investigated grokking in the modular arithmetic tasks using a one-layer transformer, and found that Boltzmann entropy (logarithm of solution volume) as an excellent indicator of generalization. By sampling the entropy landscape of the network on training loss and test accuracy, we found that i). there is no entropy barrier between the memorization state and the generalization state, which means that the memorization state is not in an equilibrium or meta-stable state. Why does grokking take so long to happen even without crossing free energy barriers? In the last part of this section, we postulate that the reason is the formation of glass states. ii). The equilibrium test accuracy, as a function of the training loss, appears continuous. This strongly suggests that that grokking is not a first-order phase transition as argued by previous work [1], because a first-order phase transition implies discontinuity of physical quantities. Our results also indicate that the equilibrium state in the entropy landscape has a significant advantage over the training curve, which we refer to it as the high-entropy advantage under grokking. Such a high-entropy advantage is weakened but still apparent when grokking is eliminated via fixing weight norm. This demonstrates a correlation between equilibrium state and NNs' generalization, consistent with the conclusion of Yang et al.[2]. In addition, we designed a toy optimizer based on the Wang-Landau molecular dynamics method, WanD, which encourages exploration of the high entropy state for NNs' training. WanD successfully achieves generalization with a similar time efficiency as AdamW but avoids grokking. The generalization states found by WanD have a much higher weight norm compared with the one needed to eliminate grokking in traditional training [3]. This further supports the conclusions of Kumar et al. [10] that grokking is not solely due to weight decay as some theory proposed before [3] in a strictly-defined context.

6.2 Discussion

Based on the above conclusions, we proposed a hypothesis: grokking is similar to the process of quickly cooling a liquid to form a glass and then slowly letting it relax back to the equilibrium state. Through this analogy, we try to answer two key questions about grokking:

What causes the NN to fall into the memorization state? One requirement for the formation of a glass is to cool the liquid below the glass transition temperature [33]. Below this temperature, the free movement of atoms is greatly restricted, thus hindering the system from evolving towards equilibrium. In grokking tasks such as modular arithmetics, the NNs are severely overparameterized, i.e., the number of the parameters is much larger than the size of the dataset. During training, the NN can fully memorize all training data and achieve a very low training loss. If we take the training loss as the energy, this is equivalent to lowering the temperature below the glass transition temperature. Another requirement for glass formation is a sufficiently fast cooling rate, so that the molecules do not have time to rearrange themselves into the lower-free-energy crystalline structure. During the training process, the gradient descent based optimizers, especially adaptive optimizer like Adam, effectively implement a rapid 'cooling' process by aggressively driving parameters towards regions of reduced training loss. This rapid descent often proceeds with insufficient perturbations (caused by large learning rate, small batch size, and data accuracy, etc.) to escape from poor local minima, making NNs easier to reside in a non-generalizable loss minima and therefore form a computational glass. In contrast, the Langevin dynamics in the WanD optimizer we designed enriches the perturbations in training, thus avoiding the formation of computational glass.

What is the internal dynamics of the grokking process? When a liquid is rapidly cooled, it can form a non-equilibrium glassy state that gradually approaches equilibrium through structural relaxation—a slow kinetic process [27]. Due to the sluggish motion of atoms at low temperatures, the relaxation time can far exceed the timescale of the cooling process. Analogously, in grokking tasks, the equilibrium state at low training loss corresponds to good generalization. However, parameters located in the low train loss regime exhibit very small gradients, which severely limits the ability of gradient-based optimizers to effectively update the network toward this generalizing equilibrium. This dynamical slowdown results in a substantial delay between the memorization and generalization phases. The effect is further exacerbated by adaptive optimizers like Adam, where the non-uniform scaling of gradients can further dampen parameter updates, increasing the likelihood of being trapped in non-generalizable minima [34]. Although the duration of the grokking process varies across

training runs, as observed in prior studies and our experiments, the transition to generalization consistently takes significantly longer than the memorization stage [8], mirroring the slow relaxation behavior characteristic of glassy systems [27].

Our conclusions agree and disagree with some previous studies. There were a series of work arguing that grokking is due to a transition from poor to good representation learning [10, 22, 23, 24]. This agrees with our analogy of relaxation from a poorly structured "glassy" (memorizing) state to a more ordered (generalizing) configuration. In fact, our entropy perspective offers a resolution to debates regarding solution "efficiency" and parameter norms in previous works [10, 22]. While Varma et al.[22] argue that grokking requires a transition to an "efficient" generalizing circuit characterized by a lower parameter norm, Kumar et al.[10] challenges it as they identified generalizing solutions with higher norm than memorizing solutions. Our framework suggests this conflict stems from using parameter norm alone to define efficiency, while entropy is actually a better indicator since the efficient solution will have more "free" parameters and therefore a larger entropy. This is also consistent with Merrill et al. [25] who showed that memorization solution and generalization solution correspond to dense and sparse subnetworks, respectively. In the sparse subnetworks, only a few neurons participate in the memorization of the rules, and the remaining degrees of freedom contribute to the entropy, so it should have a larger entropy as we observed. Kumar et al. [10] proposed that the key determinant of grokking is the rate of feature learning, which is consistent with our glass analogy, where the rate-limited free motion of particles is an important reason for the glass formation and long relaxation time. The finding by Liu et al. [24] that a fast-learning decoder can help trigger grokking also supports our analogy, as the rapid decoder acts similarly to 'fast cooling', trapping the network in a glassy memorization state. Our work also supports the finding that perturbation is beneficial to the grokking learning process [26], as WanD is able to eliminate grokking directly. Rubin et al. [1] claimed that grokking is a first-order phase transition process, but our results overturn this conclusion. This is probably because their network architecture and training data are different from ours, while ours are consistent with the original paper that discovered grokking. In addition, the high-weight norm generalization states we found through WanD also challenge the conclusions of Liu et al. [3]. Due to space limitations, we provide additional discussion in the Appendix C.

6.3 Limitation and future work

Limitation: i). **Task scope**: Our study is limited to one-layer transformers on modular arithmetic tasks and does not include other grokking benchmarks (e.g., tasks in [1, 3]). ii). **Interaction with explicit regularization**: The relationship between "glassy dynamics" and various explicit regularization methods in grokking systems remains unclear and requires further exploration. iii). **Proof-of-concept nature of WanD optimizer**: WanD has only been tested in arithmetic task scenarios, is sensitive to hyperparameters, and exhibits instability near generalization.

Our hypothesis of grokking and glass relaxation invites further investigation. While we focused on experiments with the original grokking tasks [8], future studies should examine whether our findings are applicable for other grokking tasks. Furthermore, a recent study has compared the behavior of deep neural network and structural glass with quantitative measurements [28], and we can refer to its research method to verify whether the grokking NN is a structural glass. In addition, it remains unclear how does glassy dynamics affect the effectiveness of regularization methods in removing grokking. Finally, we also aim to further improve the WanD optimizer to enhance its stability and general applicability. As an algorithm based on WLMD, WanD involves multiple hyperparameters that require tuning. Our experiments identify the learning rate, temperature, and friction coefficient as the three most influential factors affecting training outcomes. We plan to further investigate empirical guidelines for effective hyperparameter selection. Techniques from statistical physics designed to accelerate equilibration, such as simulated annealing and parallel tempering, can also be applied to WanD's future refinement [35]. We anticipate that approaches such as a dynamic hybrid optimization strategy combining WanD and AdamW could help mitigate instability issues and present some preliminary results in the Appendix B.3. Ultimately, we aim to develop WanD into a general-purpose optimizer applicable to a broader range of machine learning tasks.

Acknowledgments

The authors thank National Natural Science Foundation of China for supporting this research (Grant 12405043, G.Z.).We also thank computational resources provided by Bridges-2 at Pittsburgh Supercomputing Center through ACCESS allocation CIS230096 (E.Y.).

References

[1] Noa Rubin, Inbar Seroussi, and Zohar Ringel. Grokking as a first order phase transition in two layer networks. *arXiv preprint arXiv:2310.03789*, 2023.

[2] Entao Yang, Xiaotian Zhang, Yue Shang, and Ge Zhang. High-entropy advantage in neural networks' generalizability. *arXiv preprint arXiv:2503.13145*, 2025.

[3] Ziming Liu, Eric J Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data. *arXiv preprint arXiv:2210.01117*, 2022.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[8] 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.

[9] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[10] Tanishq Kumar, Blake Bordelon, Samuel J Gershman, and Cengiz Pehlevan. Grokking as the transition from lazy to rich training dynamics. *arXiv preprint arXiv:2310.06110*, 2023.

[11] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pages 1309–1318. PMLR, 2018.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

[13] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[14] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR, 2017.

[15] Alexander Shevchenko and Marco Mondelli. Landscape connectivity and dropout stability of sgd solutions for over-parameterized neural networks. In *International Conference on Machine Learning*, pages 8773–8784. PMLR, 2020.

[16] Zeke Xie, Issei Sato, and Masashi Sugiyama. A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. *arXiv preprint arXiv:2002.03495*, 2020.

[17] Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

[18] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

[19] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[20] Yu Feng, Wei Zhang, and Yuhai Tu. Activity–weight duality in feed-forward neural networks reveals two co-determinants for generalization. *Nature Machine Intelligence*, 5(8):908–918, 2023.

[21] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

[22] Vikrant Varma, Rohin Shah, Zachary Kenton, János Kramár, and Ramana Kumar. Explaining grokking through circuit efficiency. *arXiv preprint arXiv:2309.02390*, 2023.

[23] Daniel Kunin, Allan Raventós, Clémentine Dominé, Feng Chen, David Klindt, Andrew Saxe, and Surya Ganguli. Get rich quick: exact solutions reveal how unbalanced initializations promote rapid feature learning. *Advances in Neural Information Processing Systems*, 37:81157–81203, 2024.

[24] Liu, Ziming and Kitouni, Ouail and Nolte, Niklas S and Michaud, Eric and Tegmark, Max and Williams, Mike. Towards understanding grokking: An effective theory of representation learning. *Advances in Neural Information Processing Systems*, 35:34651–34663, 2022.

[25] William Merrill, Nikolaos Tsilivis, and Aman Shukla. A tale of two circuits: Grokking as competition of sparse and dense subnetworks. *arXiv preprint arXiv:2303.11873*, 2023.

[26] Vimal Thilak, Etai Littwin, Shuangfei Zhai, Omid Saremi, Roni Paiss, and Joshua Susskind. The slingshot mechanism: An empirical study of adaptive optimizers and the grokking phenomenon. *arXiv preprint arXiv:2206.04817*, 2022.

[27] Birte Riechers, Lisa A Roed, Saeed Mehri, Trond S Ingebrigtsen, Tina Hecksher, Jeppe C Dyre, and Kristine Niss. Predicting nonlinear physical aging of glasses from equilibrium relaxation via the material time. *Science advances*, 8(11):eabl9809, 2022.

[28] Max Kerr Winter and Liesbeth MC Janssen. Glassy dynamics in deep neural networks: A structural comparison. *Physical Review Research*, 7(2):023010, 2025.

[29] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

[30] Christoph Junghans, Danny Perez, and Thomas Vogel. Molecular dynamics in the multicanonical ensemble: Equivalence of wang–landau sampling, statistical temperature molecular dynamics, and metadynamics. *Journal of chemical theory and computation*, 10(5):1843–1847, 2014.

[31] Kim, Jaegil and Straub, John E and Keyes, Thomas. Statistical-temperature Monte Carlo and molecular dynamics algorithms. *Physical review letters*, 97(5): 050601, 2006.

[32] Fort, Gersende and Jourdain, Benjamin and Kuhn, Estelle and Lelièvre, Tony and Stoltz, Gabriel. Convergence of the Wang-Landau algorithm. *Mathematics of Computation*, 2015.

[33] Jean-Pierre Hansen and Ian Ranald McDonald. *Theory of simple liquids: with applications to soft matter*. Academic press, 2013.

[34] Zhou, Pan and Feng, Jiashi and Ma, Chao and Xiong, Caiming and Hoi, Steven Chu Hong and others. Towards theoretically understanding why sgd generalizes better than adam in deep learning. *Advances in Neural Information Processing Systems*, 33, 21285-21296, 2020.

[35] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*. Elsevier, 2023.

[36] Chaudhari, Pratik and Choromanska, Anna and Soatto, Stefano and LeCun, Yann and Baldassi, Carlo and Borgs, Christian and Chayes, Jennifer and Sagun, Levent and Zecchina, Riccardo. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We listed and described claims made in this work both in the abstract and the introduction. The main claims are made based on our experimental results (section 4.1, 4.2 and 5).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed the limitations of our work and future directions in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We propose a computational glass relaxation hypothesis based on the observed entropy landscape in grokking tasks and demonstrate the validity of the analogy. We also verify this result using an optimizer we designed, WanD.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We give the sampling method in the Section 3 and provide the hyperparameters required for our work in the Appendix. Readers can reproduce the results based on the code we provide.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide code about i). Sampling entropy using WLMD ii). The optimizer WanD we designed.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide experimental details in the Appendix. Full details are given in the code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Due to the computational cost of entropy landscape sampling, we fixed the same data split and random seed for both training and sampling procedures, so error analysis is not applicable.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We give the hardware environment used in the experiment and the computational cost in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work of grokking conform with the NeurIPS Code of Ethics in every respect.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our work towards understanding generalization of grokking is of academic interest only and does not have direct societal impact.

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work on grokking in modular arithmetic tasks do not include such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We give proper credit to the assets we used mainly via citations and textual descriptions.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We currently do not introduce new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: We do not involve LLMs in this work.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Additional Experimental Details

A.1 Model and Datasets

We followed the work of Nanda et al.[29] for the model setup of modular arithmetic tasks. We use the same one-layer transformer model for the programs of AdamW optimizer training, WanD optimizer training, and WLMD entropy sampling. This transformer model has 4 attention heads by default, each with a width of 32, yielding a model dimension of 128, and attached with an MLP layer with a width of 512. In the experiment of eliminating grokking by controlling the weight norm, we modified the attention head width (8/16/32) and the MLP layer width (128/256/512). In order to reduce the computational cost of WLMD entropy sampling, we selected a smaller prime number 67 as the modulus for our task. Although some works [3, 29] choose 113 as the modulus, there is still exist grokking in arithmetic tasks with 67 as the modulus, so we think this setting is appropriate. We also use the cross-entropy loss as our loss function. For more details on the model's hyperparameters, please refer to the provided code.

A.2 Training Details and Cost

For modular arithmetic tasks training by AdamW, we use a full batch size on 1 GeForce RTX 4090 GPU, training for 10000 optimization steps with weight decay 1 or fixed weight norm 30. The hyperparameters for training with AdamW are given in Table 1. For our toy optimizer WanD training, since it is not yet fully developed, there are many hyperparameters that directly affect the generalization of the training results. Among them, the learning rate, temperature, and friction coefficient are the three key factors that determine the training results. We provide a set of hyperparameter combinations in the code that can generalized well.

Table 1: AdamW training hyperparameters

Operation	Model Dimension	MLP Width	Learning Rate	Weight Decay	Weight Norm
$x+y$	32	128	1e-3	0	Fixed at 30
$x+y$	64	256	1e-3	0	Fixed at 30
$x+y$	128	512	1e-3	0	Fixed at 30
$x+y$	128	512	3e-3	1	Not fixed
x^2+y	128	512	5e-3	1	Not fixed
x^3+xy^2+y	128	512	1e-2	1	Not fixed

For WLMD sampling, we use 4 GPUs to run 8 processes. Each process uses the same data split and updates the entropy landscape according to a different random seed. A single process runs 100,000 epochs at a time, then calculates the average entropy of all processes and updates the entropy landscape. When running 8 WLMD processes on 4 GeForce RTX 4090 GPUs, it takes about a day for each process to update 1M epochs. When running 8 WLMD processes on 4 NVIDIA H100 GPUs, it takes about 16 to 17 hours for each process to update 1M epochs. In order for the entropy landscape to converge to its true distribution, tens to hundreds of millions of epochs are typically required. For detailed hyperparameter settings, please refer to the code we provide.

A.3 WLMD details

Smoothed accuracy: To ensure well-defined gradient of test accuracy with respect to NN parameters in WLMD, we compute a smoothed test accuracy using a sigmoid function, following [2]. The detailed procedure for obtaining smoothed accuracy is outlined in Algorithm 2. In our work, we choose the smoothed accuracy slope $s = 7$.

Entropy bias: When sampling the entropy landscape of grokking NNs, we are interested in the regime with lower training loss, which corresponds to the memorization state and the generalization state. However, areas with high training loss usually have higher entropy (as evidenced by randomly initialized networks that generally possess high training loss), which will waste a lot of time exploring in the early stages of sampling. In order to prevent the simulation program from wasting time outside the range of our interest, we set an additional entropy bias:

$$\text{if } ln(L_{train}) > L_0,\ S_{bias}(ln(L_{train})) = \beta(ln(L_{train}) - L_0)^2;\ \text{else } S_{bias}(ln(L_{train})) = 0. \quad (7)$$

Algorithm 2 Smoothed Accuracy

Require: Transformer model $T(\cdot)$, dataset $(X = \{x_1, x_2, ..., x_n\}, \hat{Y} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\})$, smoothed accuracy slope s

Ensure: The value of smoothed accuracy close to the true accuracy, and the gradient of smoothed accuracy with respect to parameters has a finite value.

for each $(x_i, \hat{y}_i)$ in $(X, \hat{Y})$ **do**

Get the highest logit among all wrong output classes: $p_i(y = y_{i\_wrong})$ where $y_{i\_wrong} \in Y_i = T(x_i)$

Get the logit of the correct output class: $p_i(y = \hat{y}_i)$

Compute the smoothed accuracy:$A_i = Sigmoid((p_i(y = \hat{y}_i) - p_i(y = y_{i\_wrong})) \cdot s)$

end for

Compute the total smoothed accuracy:$A = \sum A_i/n$

If the training loss is outside the range of interest ($ln(L_{train}) > L_0$) when updating the NN parameters, this entropy bias will provide additional entropy gradients to to force the NN to move towards lower training loss.

The overall WLMD entropy sampling procedure is outlined in Algorithm 3. We employ the cross-entropy loss for our modular arithmetic task and a logarithmic scale of the training loss is used to better examine the entropy landscape near zero training loss.

Algorithm 3 WLMD Entropy Sampling Method

Require: Transformer model $T(\cdot)$, its current entropy landscape S, the entropy bias S_{bias}, training data $(X_{train}, y_{train})$and test data(X_{test}, y_{test});

Ensure: Convergent entropy landscape $S(x_{WL}, y_{WL})$ where $x_{WL} = ln(L_{train})$, $y_{WL} = A_{test}$;

Initialize: $S \leftarrow S_{bias}$;

for each sampling epoch **do**

Compute the logarithm of the train loss and smoothed test accuracy:
$ln(L_{train}) = ln(CrossEntropyLoss(T(X_{train}), y_{train}))$;
$A_{test} = SmoothedAccuracy(T(X_{test}), y_{test})$;

Compute entropy gradient $x_{grad} = \frac{\partial S(x_{WL}, y_{WL})}{\partial x_{WL}}$ and $y_{grad} = \frac{\partial S(x_{WL}, y_{WL})}{\partial y_{WL}}$;

The combined loss $L_{combined} = -x_{WL}x_{grad} - y_{WL}y_{grad}$;

Do $L_{combined}.backward()$ and get $\frac{d^2c_i}{dt^2}$;

for each parameter c_i **do**

Update the parameter velocity by Equation 5: $\frac{dc_i}{dt} \leftarrow \frac{dc_i}{dt}(1 - c_f) + \frac{d^2c_i}{dt^2}\Delta t + v_{Langevin}$;

Update the parameter: $c_i \leftarrow c_i + \frac{dc_i}{dt}\Delta t$;

end for

Update the Wang-Landau factor by Equation 3: $f_{WL} \leftarrow F_{WL}min\{t/t_{WL}, t_{WL}/t\}$;

Update the entropy landscape by Equation 3: $S(x_{WL}, y_{WL}) \leftarrow S(x_{WL}, y_{WL}) + f_{WL}\delta(x - x_{WL}, y - y_{WL})$;

end for

$S \leftarrow S - S_{bias}$;

B Additional Results

B.1 Visualization of modular arithmetic tasks

First, we show a visualization of the three modular arithmetic tasks in the Figure 4. A complete dataset with 67*67=4489 data points can be viewed as a 67*67 pixel image, where the x-axis and y-axis coordinates of the pixels correspond to the two numbers $\langle x \rangle$ and $\langle y \rangle$ involved in the operation, and the color of the pixel corresponds to the answer $\langle x \circ y \rangle$. Half of the pixels are randomly selected as the training set, and the other half as the test set. Training the transformer model to learn the modular arithmetic operation is equivalent to completing the entire image according to the regularity of the image after being masked. In particular, the numbers in the arithmetic operations are just equivalent to different characters, and they can be replaced by any other different characters without

affecting the validity of the modular arithmetic task. In order to observe the image intuitively, we dye the pixels with light to dark blue according to the corresponding numbers from small to large. It is not difficult to observe that the visualization images of simpler tasks have more obvious regularities. For the task that cannot be learned, its visualization image is close to random noise.

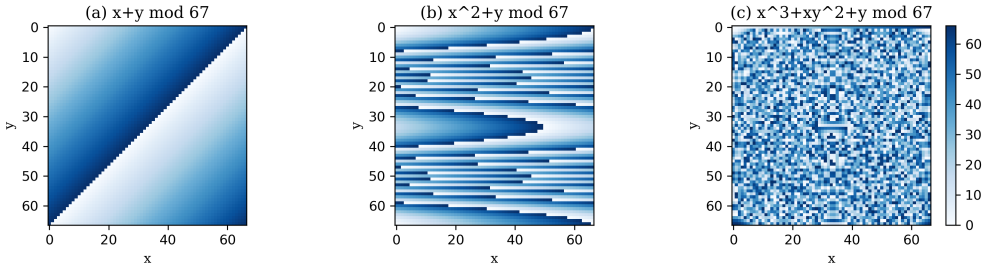

Figure 4: Visualization of three different arithmetic tasks. (a) $x + y \ mod \ 67$. (b) $x^2 + y \ mod \ 67$. (c) $x^3 + xy^2 + y \ mod \ 67$. Our protocol is equivalent to randomly taking 50% of the pixels from the image as the training set and trying to complete the rest. The more complex the expression, the more visually irregular the image is.

B.2 AdamW training results of modular arithmetic tasks

The AdamW training results of these three modular arithmetic tasks are shown in Figure 5. The figure shows that as the expression becomes more and more complex, the learning difficulty of the NN also increases, from easy to learn, to difficult to learn, to impossible to learn. The generalization delay is more significant in the difficult task, which means grokking is more pronounced.

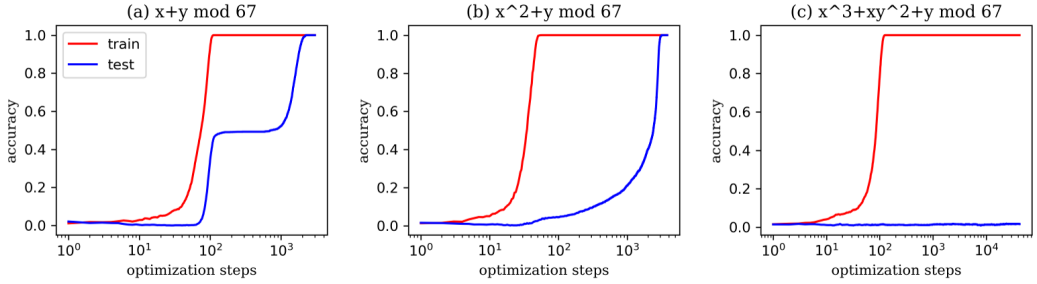

Figure 5: Training results of three different arithmetic tasks: (a) $x + y \ mod \ 67$ is easy to learn, but the hysteresis of the test accuracy shows grokking. (b) $x^2 + y \ mod \ 67$ is hard to learn, the grokking is more obvious. (c) $x^3 + xy^2 + y \ mod \ 67$ cannot be learned.

We also train the transformer in the modular addition task with a constrained weight norm. The weight norm is restricted to the Goldilocks zone [3], and we set it to 30 for three NNs with different widths. We tested three NNs with different widths mainly because Yang et al. [2] observed that the high-entropy advantage becomes more significant with smaller width. The training results are shown in Figure 6. In fact, these three NNs reach generalization after almost the same number of optimization steps, and none of them exhibits grokking. Combined with the entropy landscape in Figure 2 in the main text, we believe that the relationship between high-entropy advantage and NN width is not obvious in the grokking task.

B.3 Training results of AdamW and WanD under different dataset splits

Increasing the training dataset split fraction is an effective way to eliminate grokking [8]. We tested the training results of AdamW and WanD on the $x + y \ mod \ 67$ (Figure 7) and $x^2 + y \ mod \ 67$ (Figure

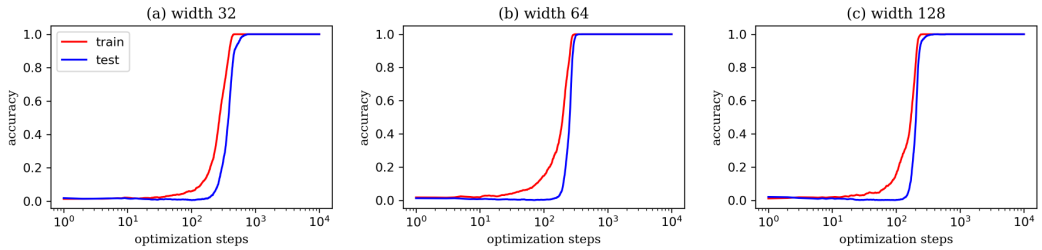

Figure 6: Training results of modular addition ($p = 67$) task with three transformer models that have different widths. Here weight norm is constrained to a fixed value of 30.

8) tasks at the same learning rate 0.003, with the training dataset fraction being 50%, 60%, 70%, and 80%, respectively. Here we anneal WanD with AdamW so that the update weight of WanD increases first and then decreases during training, thus avoiding the instability problem at the end of WanD training. Specifically, in each iteration, the NN parameters are updated by the WanD optimizer with weight α and by the AdamW optimizer with weight $1 - \alpha$, where $\alpha = (2A_{test} - 1)^2$.

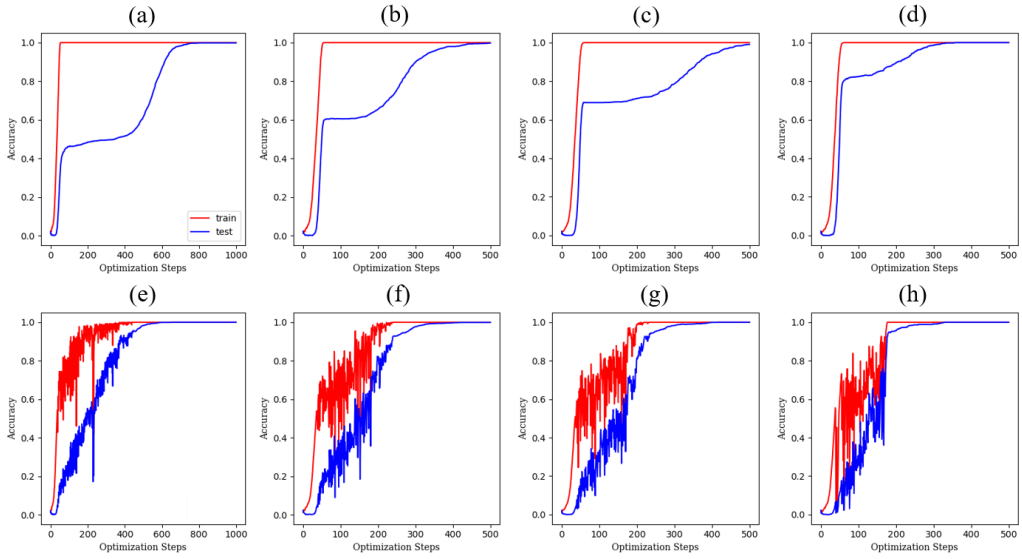

Figure 7: AdamW and WanD training results of $x + y \ mod \ 67$ task with different training dataset fraction. Top row:AdamW; bottom row:WanD; (a)(e) 50%, (b)(f) 60%, (c)(g) 70%, and (d)(h) 80% training dataset split fraction. The hyperparameter of WanD: kT=0.5, frictionCoefficient=0.5, maxTrainLossToStudy=-8.

The results show that WanD has a significant training efficiency advantage in tasks with severe grokking, and even in tasks where grokking is greatly eliminated, WanD can still achieve generalization ahead of AdamW. This result demonstrates the great potential of WanD as a toy optimizer and motivates us to continue improving it.

C Additional Discussion

Robustness of entropy landscape diagrams: In this work, we visualize the entropy landscape with respect to the logarithm of the training cross-entropy loss and the smoothed test accuracy. Due to the symmetry between training and test datasets, modifying the training loss formulation is conceptually equivalent to altering the test loss. Our reported results are based on the smoothed

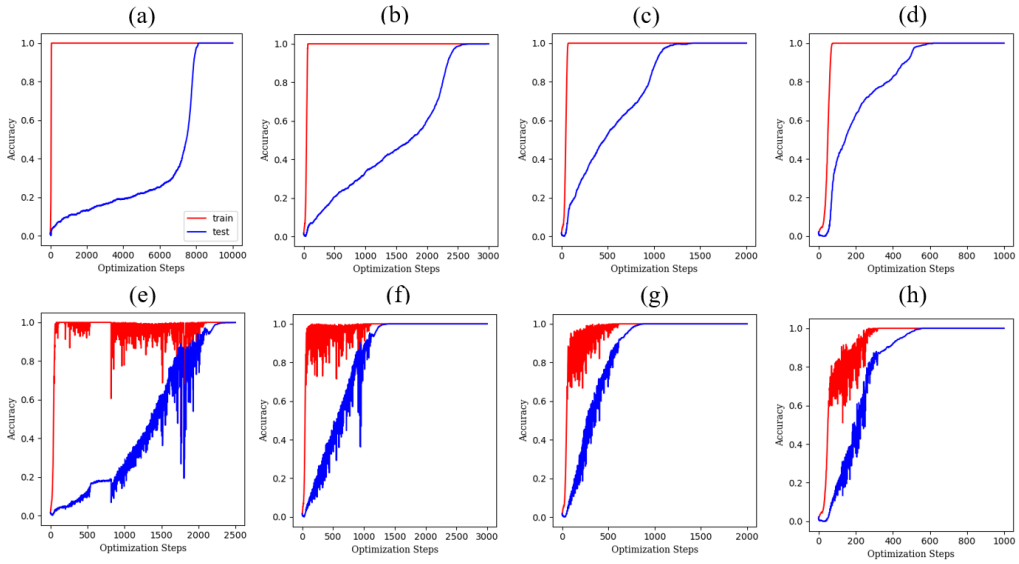

Figure 8: AdamW and WanD training results of $x^2 + y \ mod \ 67$ task with different training dataset fraction. (a)(b)(c)(d):AdamW; (e)(f)(g)(h):WanD; (a)(e)/(b)(f)/(c)(g)/(d)(h): 50%/60%/70%/80% training dataset split fraction. The hyperparameter of WanD: kT=0.7, frictionCoefficient=0.6, maxTrainLossToStudy=-8. It is worth noting that WanD in (e) achieves generalization with nearly three times the efficiency of AdamW in (a).

test accuracy, whose inverse can be interpreted as a form of loss function. We have also conducted internal experiments using the cross-entropy test loss in place of the smoothed accuracy. The resulting entropy diagram consistently support the same conclusion: entropy barrier is not observed, and only a single connected high-entropy region exists, indicating no sign of a first order phase transition. This consistency confirms that our findings are robust to the specific choice of the loss function.

Sparse subnetworks: As shown in [25], grokking is associated with sparse subnetworks in which only a subset of neurons is involved in rule memorization. From an entropy perspective, parameters outside these functional subnetworks do not affect generalization. Perturbing such parameters preserves model performance, implying that the corresponding solution set occupies a larger volume in parameter space. Since entropy is defined as the logarithm of this volume, neural networks with sparse subnetworks exhibit higher entropy.

Other entropy-based optimizer: WanD is an optimizer built on the one-dimensional WLMD framework. It excels at identifying high-entropy states, particularly the generalized solution, via molecular dynamics, leveraging the fact that generalized solutions possess higher entropy at low training loss. Prior to our work, there is another entropy-guided optimizers named Entropy-SGD [36] have been proposed. Below, we briefly highlight the theoretical distinctions between WanD and Entropy-SGD:

- **Different notions of entropy.** The core distinction lies in the entropy definition: WanD targets the global Boltzmann entropy, reflecting the total volume of the solution space, whereas Entropy-SGD focuses on local entropy, which captures the flatness of individual minima.
- **Different methodologies.** WanD estimates entropy via 1D WLMD sampling, while Entropy-SGD relies on Langevin dynamics to approximate local entropy gradients.

Optimizers like Adam often progress slowly due to limited gradient signals during the memorization-to-generalization transition [10]. In principle, Entropy-SGD could potentially accelerate grokking if its identified high local-entropy states exhibit strong generalization. A rigorous empirical comparison between these approaches remains a valuable direction for future work.