Final Report: An Investigation of grokking phenomenon

解晟平 陈全 何雨桐

Abstract

Grokking has been a widely discussed phenomenon in machine learning research, which reflects that the generalization of model can happen far after memorization or overfitting. In this report, we based on the original paper Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets and mainly investigate this phenomenon by learning modular addition task: $(x, y) \to (x + y)\ mod\ p$. Since the original paper's code is outdated, we have referenced the following reproduction of grokking: https://github.com/Sea-Snell/grokking as the framework for our code, then we add our own implementation to study the effects of different factors, including model structure, optimizers, number of elements, and we use wandb to visualize our results. Furthermore, we discuss three different explanations of the grokking phenomenon, validate them with our own experiments and find that a sharpness-aware minimization algorithm called AdaSAM can effectively reduce the grokking delay. All the code and supplementary materials are available at: https://github.com/Tsokarsic/miml-project-3.

1 Problem setting

Before we show our results ,we first define the problem formally. Assume p is a prime number, what we should do is to train a modular addition calculator using neural network. For the inputs, as the model shouldn't get access to the integer structure during training, we first transfer both the integers$(0, 1, \ldots, p-1)$ and the operands$(+, =)$ to one-hot vectors, where we assign"+" as p and "=" as $p+1$, so the dimension of each vector is $p+2$ .Assume there are n elements to be added, the equation will be "$a_1 + a_2 + \ldots + a_n =$", and the output should be a number in $(0, 1, \ldots, p-1)$. Therefore, the model can be regarded as a classifier. For majority tasks, we selected $p = 97$ and $n = 2$.

For the model structure, we used a 2-layer multi-head transformer decoder model which consists 4 transformer heads and intermediate-dimension=512. For majority of our experiments, we use AdamW with batch size=512, learning rate=10^{-3}, Weight decay=1, $\alpha = 0.9, \beta = 0.98$ as optimizer, with linear learning rate warm-up in first 10 steps and a maximal optimization budget of 10^6 steps. Due to limitation of computation resources, we decided the end the training process if the validation accuracy maintained at over 98%.

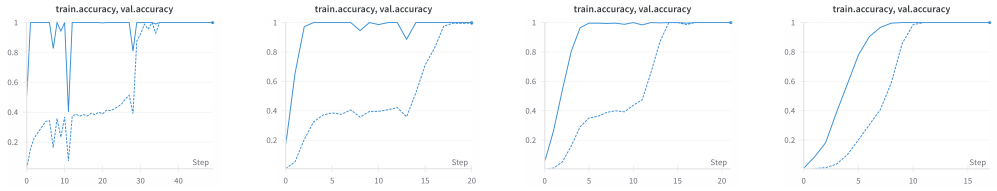
38th Conference on Neural Information Processing Systems (NeurIPS 2024).

2 Experiments

2.1 Grokking in transformer

In this section, we reproduce the results in [1]. To be specific, we show the grokking phenomenon of modular addition using the model of transformer and the optimizer of AdamW.

We first investigate the grokking phenomenon when $\alpha = 0.4$. We make experiments with p ranging in $47, 97, 149, 197$. As is shown in Figure 1, the grokking phenomenon is clear, as the validation accuracy reaches 1 long after the training accuracy reaches 1. Beyond the original article, we discovered an interesting "double-grokking" phenomenon: when we choose α=0.4 we found that the validation accuracy soon reached around 40%, and then stayed at that level for a long time until the model finally grok to near 100% accuracy.



(a) training and validation accuracy when p=47 α=0.4

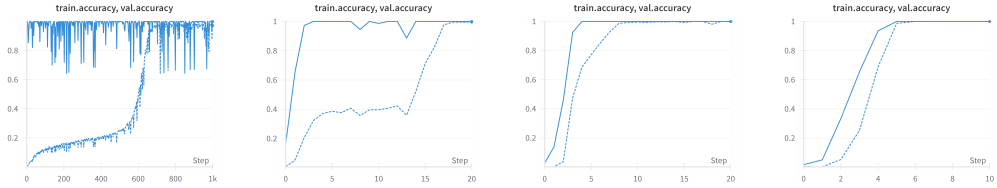
(b) training and validation accuracy when p=97 α=0.4

(c) training and validation accuracy when p=149 α=0.4

(d) training and validation accuracy when p=197 α=0.4

Figure 1: Grokking phenomenon with different p. In the diagram, we use solid lines to represent training accuracy and dashed lines to represent validation accuracy. Each step consists of 100 iterations. The following experimental results adopt this setting as well.

Meanwhile, when we compare the results using different p, we surprisingly find out that as p increases, the training time decreases, which is against our intuition. We infer this phenomena that as the model's parameters remain unchanged, It becomes more difficult for the model to fit the training data solely through memorization rather than understanding when the size of the training dataset increases, which prompt the model to lean more towards finding a solution with stronger generalization ability at the outset. We also found that as p decreases, the accuracy curve became more fluctuating and sharp.



(a) training and validation accuracy when p=97 α=0.2

(b) training and validation accuracy when p=97 α=0.4

(c) training and validation accuracy when p=97 α=0.6

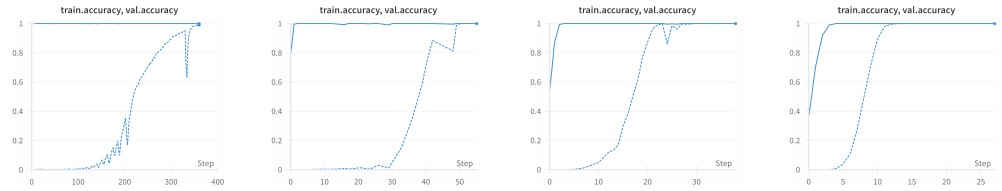
(d) training and validation accuracy when p=97 α=0.8

Figure 2: Grokking phenomenon with different training fraction α

Then we change α to examine the impact of grokking phenomenon caused by the change of α. We observe that as α increases, the grokking time decreases, which means the validation accuracy reaches 1 earlier. This is consistent with the intuition that the more training data, the easier the training process will be. The result are shown in Figure 2.

2.2 Grokking in different models

In this section we applied other models to investigate whether grokking phenomenon still exists. Specifically, we adopt MLP and LSTM to show the results. Our MLP model consists of 2 layers, $512 \rightarrow 512 \rightarrow 99$. The learning rate here is set as $5e-4$ to get more stable results. We can see from Figure 3. that grokking still exists in MLP models. We also found that when the training fraction is low, MLP model generalizes twice faster than transformer if we use training steps as standard, and if we consider the total training time, MLP model outweighs transformer more,showing MLP is probably more suitable on solving easy classification problems.



(a) training and validation accuracy when p=97 α=0.2

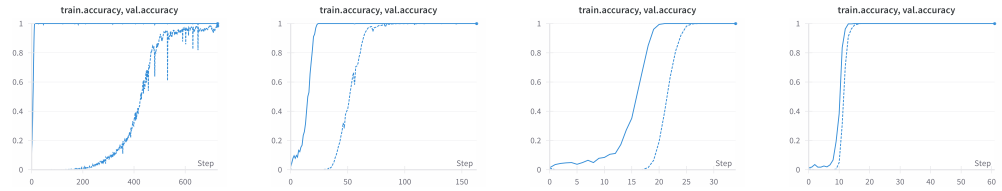
(b) training and validation accuracy when p=97 α=0.4

(c) training and validation accuracy when p=97 α=0.6

(d) training and validation accuracy when p=97 α=0.8

Figure 3: Grokking phenomenon in MLP model.

The LSTM model we adopt is a 1-depth layer with intermediate dimension 512. In our LSTM model, as shown in Figure 4, grokking phenomenon also exists. The phenomenon is similar to what is shown in the transformer model. However, we can see that the convergence time in both models is longer than that in the transformer model, which may show the superiority of the transformer structure. What is more, the relationship between α and grokking time still holds, i.e. as α increases, grokking time decreases.



(a) training and validation accuracy when p=97 α=0.2

(b) training and validation accuracy when p=97 α=0.4

(c) training and validation accuracy when p=97 α=0.6

(d) training and validation accuracy when p=97 α=0.8

Figure 4: Grokking phenomenon in LSTM model

2.3 Effects of different optimizer and regularization techniques

In this section, we will discover how different optimizers and regularization methods affect grokking phenomenon. We measure the ability of grokking by the best validation rate after 10^5 optimization steps. We selected the following optimizers: For SGD-based optimizers,we selected vanilla SGD, SGD with heavy ball momentum and SGD with Nesterov momentum, and we selected momentum 0.99 and learning rate 0.01 in order to fit the optimizer feature. For Adam,we test across different regularization methods and batch-size. We tested mini-batch Adam with batch size=128, vanilla Adam, AdamW with different weight decay rate(0.1 and 1) and Adam with Dropout 0.1. The results are shown in the following figure.

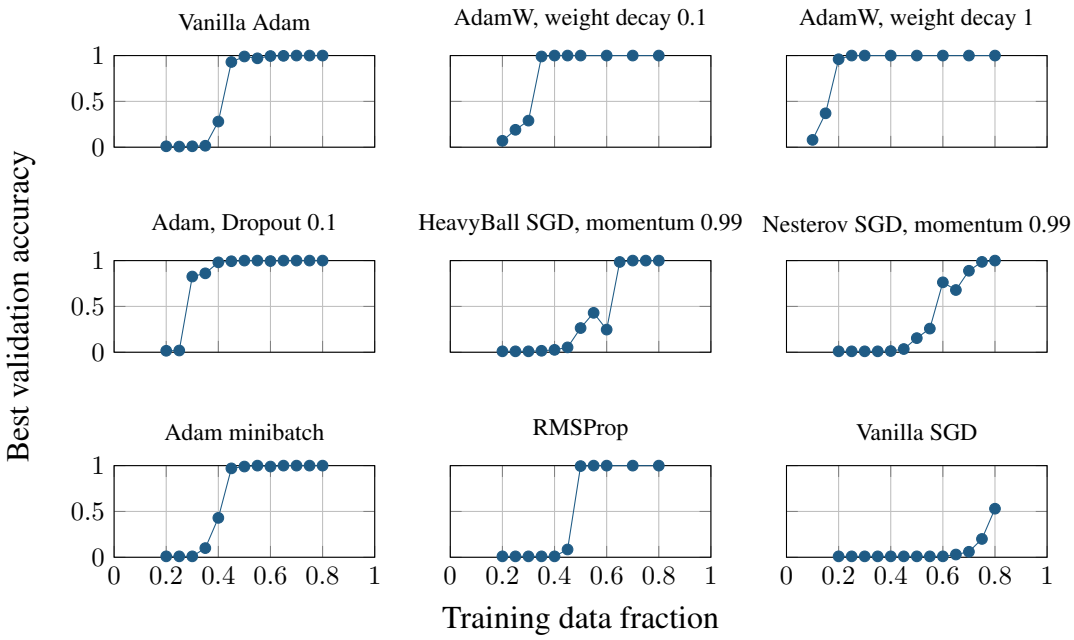


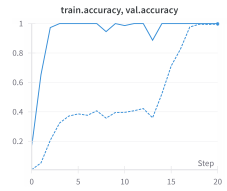
Figure 5: best validication rate of different training configurations

Figure 5 show results similar to [3][Figure 3, Section 3.3]. Here are our conclusions. Firstly, AdamW optimizers with an appropriate weight decay outperformed other regularization methods significantly, especially works when the training fraction is small， while other methods such as dropout also increase the capability of generalization. For different optimizers, Vanilla SGD without any momentum struggle to generalize even in large training fraction cases, while momentum based-SGD algorithm and RMSProp can only generalize when the training fraction is relatively large. Finally,we found using small batch-size can slightly enhance generalization when training fraction is small.

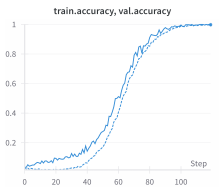
2.4 Effects of different number of summands

In this section, we discovered the effects to grokking with different number of summands, which we take different n defined in problem setting part. In order to reduce the influence of other factors and establish an appropriate unified criterion, we select different p for different n, which we choose $p = 97$ when $n = 2$, $p = 23$ when $n = 3$, $p = 11$ when $n = 4, 5$, than the dataset size when $p = 2, 3, 4$ are nearly the same, we keep other settings same as the default one in problem settings. To make the influence more clear, we evaluated the effects both on MLP models and transformer models. Our results are shown in Figure 6.

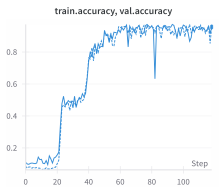
We conclude our observation based on Figure 6. First of all, when dealing with larger n, the cracking phenomenon in MLP models are still clear, but the gap between generalization and memorization are getting smaller when n grows. While in the transformer model, when n is larger, training and validation accuracy are growing consistently, furthermore, the model fails to perfectly fit $n = 5, p = 11$ cases as the training accuracy floats around 95%. For the total step to generalize, the transformer models always spend more time as n grows, while MLP models spend the most when $n = 3$, we infer that this may result from the fact that when n grows, the input data are getting more related to each other, which may improve MLP to predict but may mislead the transformer's self-attention process.



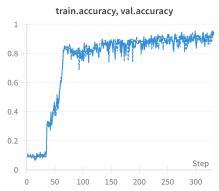
(a) grokking in transformer when p=97 n=2



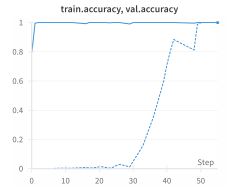
(b) grokking in transformer when p=23 n=3



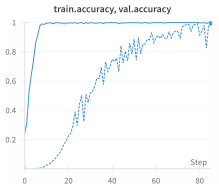
(c) grokking in transformer when p=11 n=4



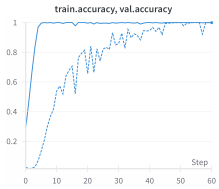
(d) grokking in transformer when p=11 n=5



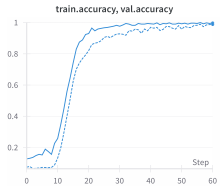
(e) grokking in MLP when p=97 n=2



(f) grokking in MLP when p=23 n=3



(g) grokking in MLP when p=11 n=4



(h) grokking in MLP when p=11 n=5

Figure 6: Grokking phenomenon with different number of summands

3 Explanation to grokking

In this section, we discuss three different explanations to the grokking phenomenon. We first discuss the explanation that minima with smaller ℓ_2 or ℓ_∞ norms may generalize better, however,our experiments show that the alignment between parameter norms and generalization ability is not as good as expected， so we further discuss another possible explanation that flatter minima may generalize better and validate by experiments that a sharpness-aware minimization algorithm, AdaSAM, can effectively solve the grokking phenomenon, beating AdamW by a large margin. Finally, we explore an idea that non-linearity may help with the grokking phenomenon, where we observe positive results by replacing ReLU activations by $\text{ReLU}^{1.5}$ or ReLU^2.

3.1 Minima with smaller norm generalize better

In machine learning tasks like the modular addition, the model to train is usually over-parameterized, meaning there are many different solutions that minimize training loss. However, not all of the minima can generalize well to out-of-distribution tasks. Consequently, when the training accuracy first becomes 100%, it is possible that the minimum that the optimization algorithm finds does not generalize well and thus leads to a poor validation accuracy. As the optimization procedure continues, the model parameters can finally tend to minima with certain properties which generalize well, and the validation accuracy finally catches up with the training accuracy.

For example, adding regularization terms, *e.g.*, ℓ_2 or ℓ_∞-norm of the training parameters to the loss function is believed to improve the generalization property. A classical theoretical result states that gradient descent implicitly bias the optimization result to be the minimum with the smallest ℓ_2-norm when solving linear regression problems. In such cases, the training accuracy may rise quickly due to over-parameterization, and after some delay, the validation accuracy catches up because the parameters finally approach the minimum with the smallest ℓ_2-norm, thus grokking occurs. Empirically, the experimental results in the previous section have shown that AdamW performs the best to reduce the grokking delay, probably because the weight decay procedure tends to find small ℓ_2-norm solutions.

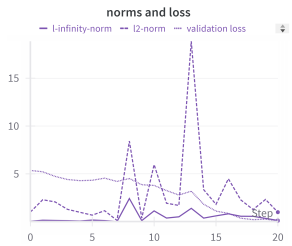


Figure 7: ℓ_2-norm, ℓ_∞-norm and validation loss when p = 97 α = 0.4

However，Theoretically, the above results may not hold for more complicated models and optimization algorithms, and we also observe in our experiments that though the decrease in validation loss has a similar trend with the parameters ℓ_2 or ℓ_∞ norm, they may not align well enough, as shown in Figure 7.

3.2 Flatter minima generalize better

A line of work on generalization property believes flat minima generalize better than sharp ones [2]. Sharpness-Aware Minimization (SAM) [1] finds flatter minima by modifying the objective function, *e.g.*, from $\min_x f(x)$ to $\min_x \max_{\|\epsilon\|_2 \leq \rho} f(x+\epsilon)$. In order to study the effect of SAM algorithms in accelerating generalization and make fair comparison with AdamW, we choose AdaSAM[5], a SAM algorithm with adaptive learning rate and momentum acceleration, detailed in Alg. 1.

Algorithm 1 AdaSAM

Input: Initial parameters x_0, $m_{-1} = 0$, $\hat{v}_{-1} = \epsilon^2$(a small positive scalar to avoid the denominator diminishing), base learning rate γ, neighborhood size ρ and momentum parameters β_1, β_2.

Output: Optimized parameter x_{T+1}.

for iteration $t \in \{0, 1, 2, \cdots, T-1\}$ **do**

Sample mini-batch $B = \{\xi_{t_1}, \xi_{t_2}, \cdots, \xi_{t_{|B|}}\}$;

Compute gradient $s_t = \nabla_x f_B(x) \mid_{x_t} = \frac{1}{|B|} \sum_{i \in B} \nabla f_{t_i}(x_t)$;

Compute $\delta(x_t) = \rho \frac{s_t}{\|s_t\|}$;

Compute SAM gradient $g_t = \nabla_x f_B(x) \mid_{x_t + \delta(x_t)}$;

$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$;

$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^{\odot 2}$;

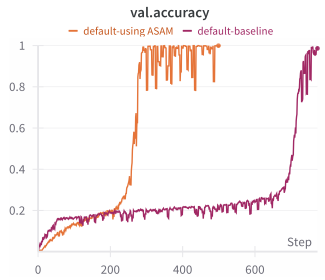
$\hat{v}_t = \max(\hat{v}_{t-1}, v_t)$;

$\eta_t = 1/\sqrt{\hat{v}_t}$;

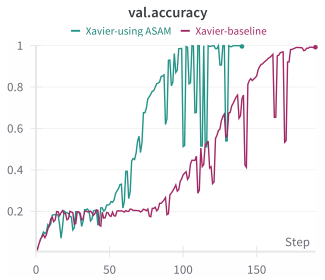
$x_{t+1} = x_t - \gamma m_t \odot \eta_t$;

end for

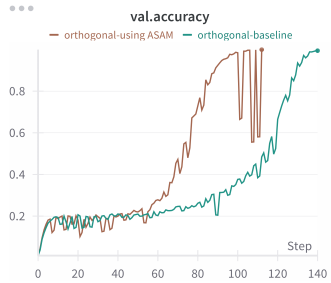
We compared AdaSAM and AdamW under different initialization settings, whose results is illustrated in Figure 8. Though using the same learning rate and momentum parameters, it is observed that under all the experimental settings, AdaSAM generalizes much faster than AdamW.Meanwhile, weight initialization methods also significantly impact the speed of grokking，probably because of the difference on landscape sharpness, and AdaSAM show more improvement in poor initialization cases, which is more robust to landscape sharpness. These results strongly indicate that flatter minima can have better generalization abilities, while minima with small ℓ_2-norms may not.



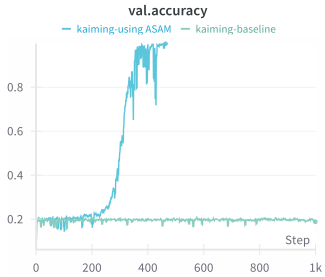
(a) Acceleration of grokking on pytorch default initialization



(b) Acceleration of grokking on xavier initialization



(c) Acceleration of grokking on orthogonal initialization

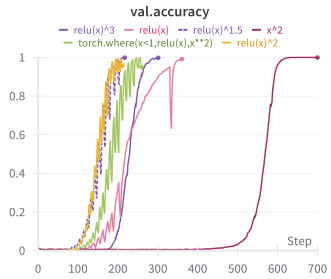


(d) Acceleration of grokking on kaiming initialization

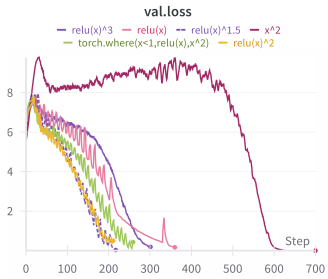
Figure 8: The effectiveness of AdaSAM optimizer on accelerating generalization and the comparison between different weight initialization methods.

3.3 Model non-linearity affects generalization

As stated in [4], the nonlinearity of the neural network, which depends on the form of the activation function and the width and depth of the network, can be employed to control the grokking behavior. Consequently, we experimented on a 2-layer MLP network to verify whether the non-linearity in the activation function can help to control the grokking behavior. As illustrated in Figure 9, by incorporating non-linearity to the ReLU activation function, the model can really generalize more quickly, where the use of $\text{ReLU}(x)^{1.5}$ and $\text{ReLU}(x)^2$ perform the best, beating the original ReLU activation by a large margin.



(a) validation acc of different activation functions



(b) validation loss of different activation functions

Figure 9: The effectiveness of activation functions on grokking in a 2-layer MLP network

Reference

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

[2] 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.

[3] 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.

[4] Ahmed Salah and David Yevick. Controlling grokking with nonlinearity and data symmetry. *arXiv preprint arXiv:2411.05353*, 2024.

[5] Hao Sun, Li Shen, Qihuang Zhong, Liang Ding, Shixiang Chen, Jingwei Sun, Jing Li, Guangzhong Sun, and Dacheng Tao. Adasam: Boosting sharpness-aware minimization with adaptive learning rate and momentum for training deep neural networks. *Neural Networks*, 169:506–519, 2024.