Final Project —— Grokking Phenomenon

Siheng Liang 2401111518
Jicong Mao 2301991812
Duo Gao 2301991693

Abstract

This Project explores "grokking" in neural networks through binary and multivariate modular arithmetic tasks, analyzing the transition from memorization to generalization and the impact of dataset size, architecture, and training methods.All the code and group information are available at： https://github.com/PKU-Lgcat/Grokking.

1 Introduction and Problem Set Up

A compelling phenomenon in neural network training is known as 'grokking' proposed by Power et al [1]. Which refers to the early stage of training in which the model exhibits memorization, that is, it achieves a very low loss in the training set but poor generalization on the test set; however, as the training time increases, the model eventually breaks out of this stage and achieves good generalization to unseen data. model is eventually able to break out of this phase and achieve good generalization to unseen data. This shift from overfitting to generalization challenges traditional machine learning theory and practice, such as early stopping strategies or criteria for terminating training based on training loss, and provokes new thinking about the inner workings of the model.

In our work, we are mainly concerned with Grokking phenomena in binary operation with small dataset, in particular the modular addition operation:

$$x \circ y = x + y \pmod{p} \tag{1}$$

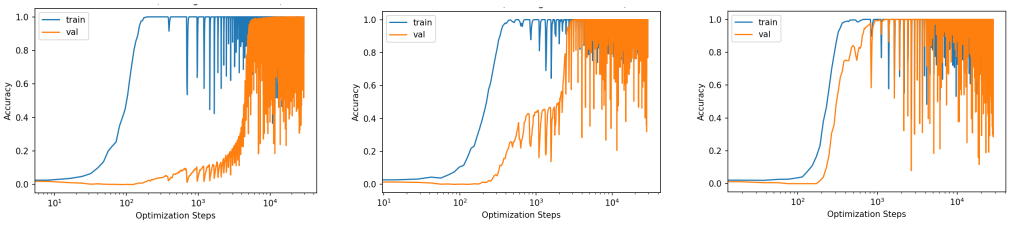
here p is a prime and $x, y \in \{0, 1, \cdots, p-1\}$.We choose $p = 41$ to ensure that a simple transformer network for solving the above problem can be trained on our local device.The original inputs are in the form of $\{(x, \text{op}, y, \text{eq}) : x, y = 0, 1, \cdots, p-1\}$, where op and eq represent for + and = in the equations, and are given value of op $= p$ and eq $= p + 1$, respectively. So for a given prime p the specific size of the original input is a matrix in $\mathbb{R}^{p^2 \times 5}$. As the experiments in original paper we use a decoder-only transformer with 2 layers, width 128, and 4 attention heads.To make the information of "integers" not known to the models, each token is transformed into a 128-dimensional embedding vector.As for the optimizer, we choose AdamW optimizer with learning rate $10^{`3}$, weight decay 1, $\beta_1 = 0.9$, $\beta_2 = 0.98$, also linear learning rate warmup over the first 10 updates starting with 0.1 times the learning rate. Since we treat this as a classification task,the cross-entropy loss function is used to evaluate the model.

2 Grokking in Different Network Architectures

2.1 Transformer

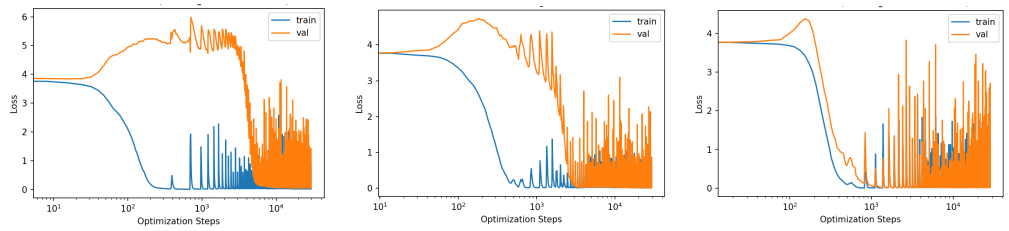
This section is about reproducing the grokking phenomenon of transform shown in the original paper but for modular addition. To be specific, we will use a transformer model with 2 layers, embedding dimension 128 , and 4 attention heads. Here we take a batchsize of 64 and a maximal optimization budget of 3×10^4 gradient updates.The result is in Figure1 and Figure2.

Preprint. Under review.



Training Fraction $\alpha = 0.3$ Training Fraction $\alpha = 0.5$ Training Fraction $\alpha = 0.7$

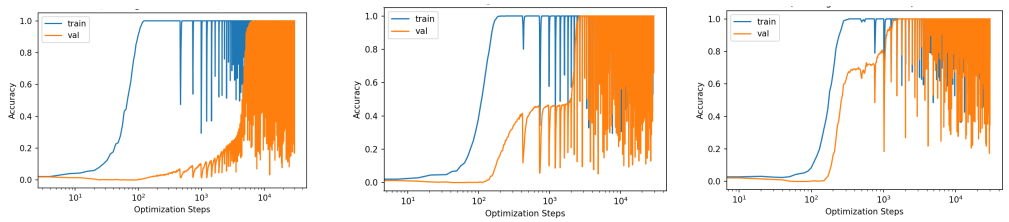
Figure 1: Accuracy of the model on the training and test sets for different training set sizes



Training Fraction $\alpha = 0.3$ Training Fraction $\alpha = 0.5$ Training Fraction $\alpha = 0.7$

Figure 2: Cross entropy Loss of the real output and model output on the training and test sets for different training set sizes

Consistent with the results in the article, the model generalizes faster when there is more data in the training set. However, the performance improvement on the test set is consistently slower than the training set.In addition to this, we set the batchsize of 128. The accuracy of the model are visualized in Figure3.



$\alpha = 0.3$ and batchsize = 128 $\alpha = 0.5$ and batchsize = 128 $\alpha = 0.7$ and batchsize = 128

Figure 3: Accuracy of the model on the training and test sets for different training set sizes with batchsize = 128

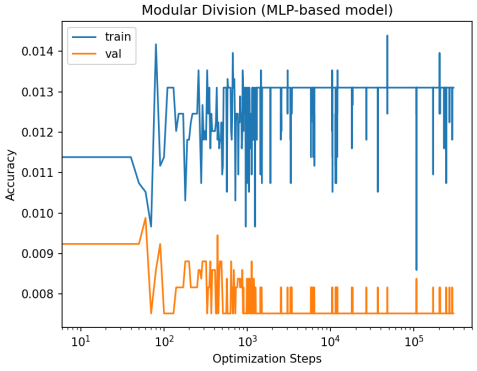
Consistent with the previous results, a larger training set size leads to better generalization performance. At the same time, we find that increasing the training batch size leads to greater volatility in the model prediction results.

2.2 MLP and LSTM

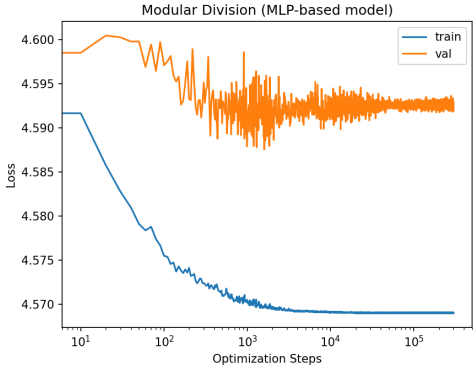
In this section we investigate the Grokking phenomenon on other neural network architectures.In this part and next part we consider modular division.

$$x \circ y = x/y \pmod{p} \tag{2}$$

We use MLP and LSTM to predict the results, and the test set accuracy and cross-entropy loss function values during training are in Figure4 and Figure5

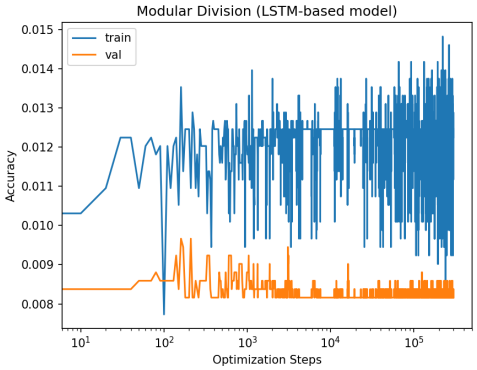


Accuracy

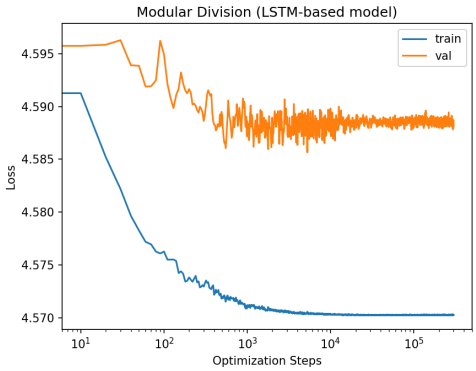


Value of Loss function

Figure 4: Grokking in MLP



Accuracy



Value of Loss function

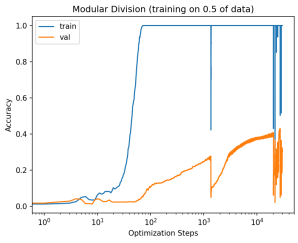
Figure 5: Grokking in LSTM

3 Impact of different optimizers and regularization methods

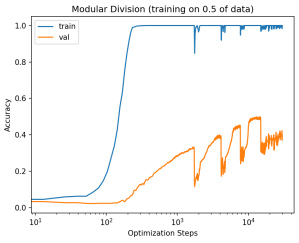
Next we will show the impact of different optimizers and regularization methods.In this section, we consider modular division.

$$x \circ y = x/y \pmod{p} \tag{3}$$

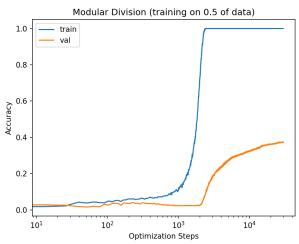
Here $x \in \{0, 1, 2, \cdots, p-1\}$ and $y \in \{1, 2, \cdots, p-1\}$ .Our modeling architecture still uses the transformer model used in the previous section to handle modular addition.For the AdamW optimizer we consider the fullbatch and minibatch AdamW and the regularization methods includes dropout and different weight decay. We also tested the SGD accelerated by Nesterov momentum.The result is shown in Figure6 and Figure7.



fullbatch AdamW optimizer

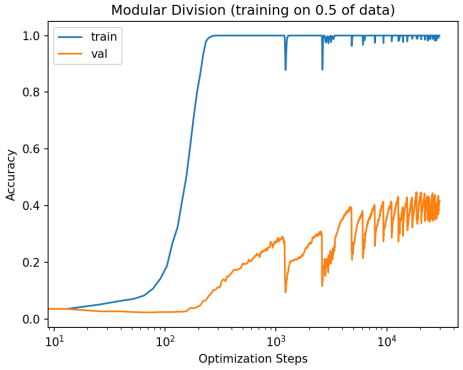


minibatch AdamW optimizer

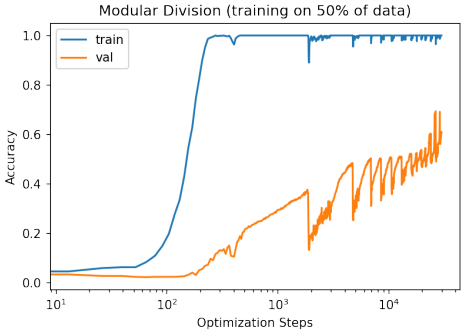


SGD with Nesterov momentum

Figure 6: Model accuracy of different optimization methods . The x-axis is the optimazation steps, and the y-axis is the best accuracy



Dropout = 0.1



Weight decay = 0.05

Figure 7: Model accuracy of different regularziation methods . The x-axis is the optimazation steps, and the y-axis is the best accuracy

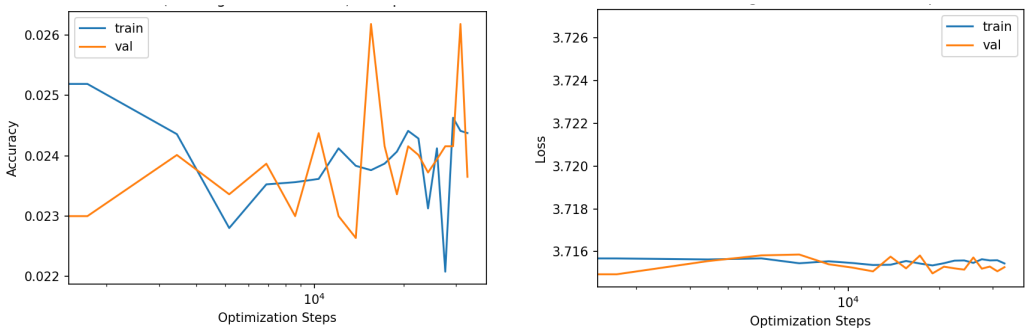
We found that for a given training budget of 3×10^4, our model can Grokking after overfitting for the binary operation of modular addition to achieve a high accuracy on the test set, but still at a low accuracy for modular division. This may be due to the size of the training set, but also due to the fact that modular division is an asymmetric operation, which is the same as the conclusion in the original paper, that for asymmetric operations, the model needs more time to generalize. At the same time, we found that various types of regularization methods will improve the performance of the model.

4 Grokking in multivariate operation Problem

In this section, we further explore the phenomenon of Grokking in multivariate operation problems,specifically, consider the following multivariate modular addition problem:

$$x_1 \circ x_2 \circ \cdots \circ x_k = x_1 + x_2 + \cdots + x_k \pmod{p} \tag{4}$$

We still uses $p = 41$ and the transformer model above.To save memory, we set $K = 3$ and batchsize = 32. We choose the training fraction $\alpha = 0.8$. The accuracy of the model on the training set as well as on the test set during the training process and the values of the cross-entropy loss function is in Figure8:



Accuracy of the model

value of the loss function

Figure 8: Accuracy of the model on the training set as well as on the test set during the training process and the values of the cross-entropy loss function in multivariate operation problem

Unfortunately,our model did not perform well on both the training and test sets even we choose α = 0.8, which may be due to the fact that the size of the data grows exponentially as K increases, and a simple Transformer network does not have enough classification power. We can still achieve high accuracy on this task by deepening the neural network, and Grokking still occurs. However, memory limitations prevent us from realizing this idea.

5 Explanation of Grokking

In this section we try to give several explanations for the phenomenon of Grokking.

5.1 Explanation through training dynamics

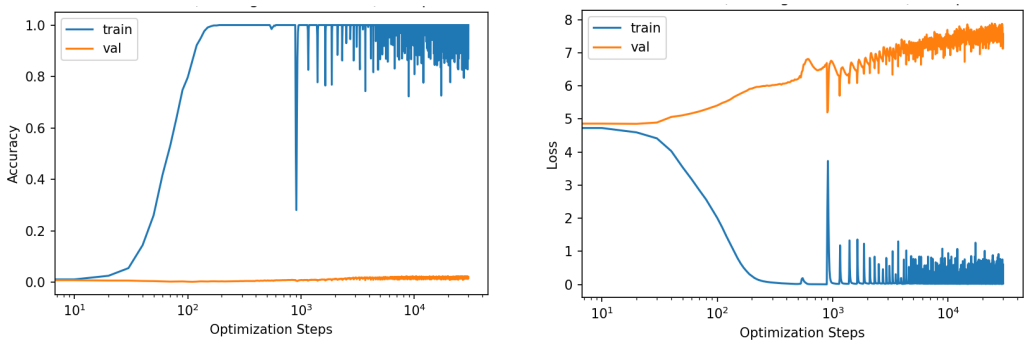
Tanishq Kumar et.al [2] give an explanation of the Grokking phenomenon through the dynamics of the training process. That is, the neural network transitions from lazy training dynamics to rich feature learning dynamics during the training process. At the beginning of training, the neural network is in the lazy training phase, which behaves like kernel regression, using initial features to fit the data; whereas, at a later stage, the network enters the feature learning phase, which leads to a better generalization solution, and delayed generalization occurs when the initial features do not agree with the objective function but the dataset is large enough to allow the network to eventually generalize. Also, the network starts in a lazy state and does not learn the features immediately, which contributes to delayed generalization.

5.2 Explanation through Linear Mapping Number

Ziming Liu et.al [3] proposed Linear Mapping Number (LMN). LMN is a new metric to measure the complexity of neural networks or their subnets. It explains the Grokking phenomenon through compression from the perspective of nonlinear complexity.LMN is a generalization of the notion of the number of linear regions in ReLU networks, and is applicable to networks with arbitrary activation functions, including smooth activation functions. It measures whether the behavior of the network is close to a linear mapping by calculating the linear connectivity between two input samples, and uses this to estimate the complexity of the network. Compared with the commonly used L2 paradigm, LMN can be more naturally interpreted as information/computational complexity, and the relationship between LMN and test loss in the compression stage is more straightforward. The authors use LMN to characterize the compression process of the network after the memory phase and before the generalization phase. The experimental results show that the LMN decreases continuously during the generalization phase and has a strong linear correlation with the test loss, while the L2 norm shows a complex nonlinear relationship.

5.3 Explanation through circuit efficiency

Circuit Efficiency was proposed by Vikrant Varma et.al [4]. They argued that neural networks have two sets of circuits both of which allow them to achieve very high accuracy on the training set. These include $\mathbf{C}_{\text{mem}}$, a circuit with strong memory capabilities, and $\mathbf{C}_{\text{gen}}$, a circuit with strong generalization capabilities. As the size of the dataset increases and the training process evolves, weight decay strengthens the advantage of $\mathbf{C}_{\text{gen}}$ over $\mathbf{C}_{\text{mem}}$, leading to delayed generalization.The following result in Figure9 demonstrates that when the training dataset is too small, the model fails to complete the delayed generalization successfully.



Accuracy of the model

value of the loss function

Figure 9: Accuracy of the model on the training set as well as on the test set during the training process and the values of the cross-entropy loss function.Here we consider the binary modular addition operation,where $p = 113$ and the training fraction $\alpha = 0.05$.We can see that Grokking don't happen.

References

[1] Alethea Power. & Yuri Burda. & Harri Edwards. & Igor Babuschkin & Vedant Misra. (2022) Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*.

[2] Tanishq Kumar. & Blake Bordelon. & Samuel J.Gershman. & Cengiz Pehlevan. (2024) Grokking as the Transition from lazy to rich training dynamics. *ICLR 2024*

[3] Ziming Liu. & Ziqian Zhong. & Max Tegmark. (2023) Grokking as Compression : A Nonlinear Complexity Perspective. *arXiv preprint arXiv:2310.05918*

[4] Vikrant Varma. & Rohin Shah. & Zachary Kenton. & Janos Kramar. & Ramana Kumar. (2023) Explaining grokking through circuit efficiency. *arXiv preprint arXiv: 2309.02390*