Final Project: Grokking Phenomenon Reproduction

Kecen Sha *
2200010611

Yuziheng Wu *
Your ID

Di Yue *
2100012961

Abstract

The abstract paragraph should be indented ½ inch (3 picas) on both the left- and right-hand margins. Use 10 point type, with a vertical spacing (leading) of 11 points. The word **Abstract** must be centered, bold, and in point size 12. Two line spaces precede the abstract. The abstract must be limited to one paragraph.

1 Introduction

Grokking [1]

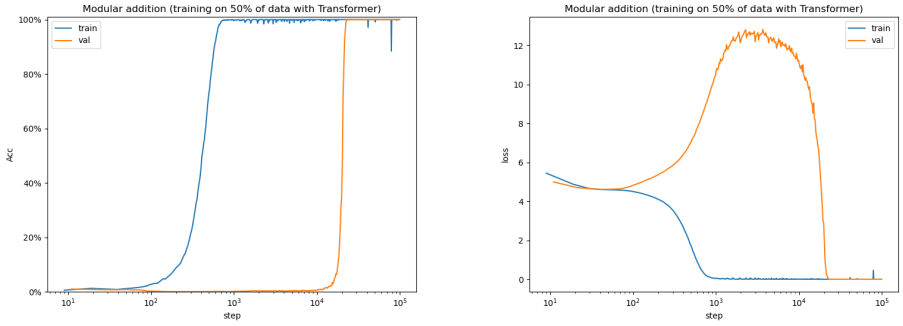
2 Preliminary

3 Experiments

3.1 Reproducing the Grokking Curve

In this task, we investigate the grokking phenomenon in modular addition with the transformer model. Specifically, we use a decoder-only transformer with 2 layers, width $d_{\text{model}} = 128$, 4 attention heads and dropout 0.1. We use the AdamW optimizer with $\beta_1 = 0.9, \beta_2 = 0.98$, learning rate 10^{-3}, weight decay 0.1, and linear learning rate warmup over the first 10 updates. For $p = 97$ and $\alpha = 0.5$, we randomly separate an α fraction from all p^2 equations in $\mathbb{Z}_p$ as the training set; the rest serves as the validation set. The model is trained with minibatch size 512 for 10^5 steps.

The accuracy and loss throughout the training process are plotted in Figure 1.



(a) Training and validation accuracy (b) Training and validation loss

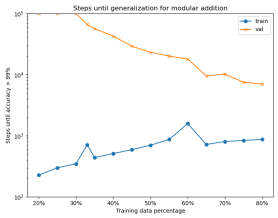
Figure 1: The grokking curves of transformer model

*Equal contribution.

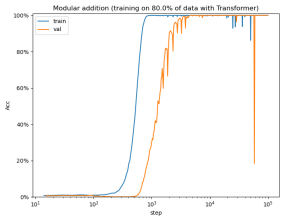
Preprint. Under review.

As is shown in Figure 1a, the model overfits the training data within 10^3 updating steps. Nevertheless, generalization does not happen until after 10^4 steps. Figure 1b further illustrates that the decrease of loss is consistent with the increase of accuracy, both for training and validation phases. Interestingly, while the training loss decreases monotonically, an increase of the validation loss is observed before it begins to converge.

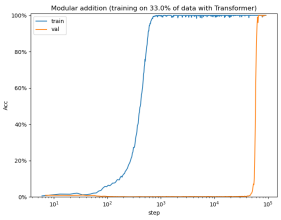
We further study the effect of the training data fraction α. For each α we sample, we train the model on a random α fraction of all p^2 equations for at most 10^5 steps, and determine the minimum number of steps required to achieve validation accuracy $\geq 99\%$. The results are plotted in Figure 2a. As a comparison, we plot the accuracy curves for $\alpha = 80\%$ and $\alpha = 33\%$ in Figures 2b and 2c, respectively.



(a) Minimal number of training steps required to achieve validation accuracy $\geq 99\%$



(b) The accuracy curve for $\alpha = 80\%$



(c) The accuracy curve for $\alpha = 33\%$

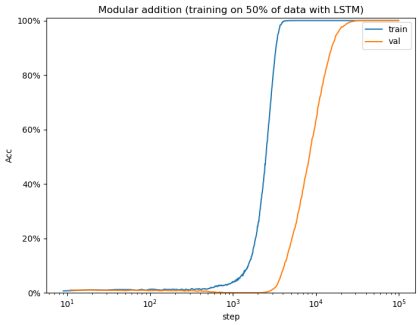
Figure 2: Effect of the training data fraction α

When $\alpha = 80\%$, the model generalizes in 10^4 steps. As α decreases, it becomes easier for the model to overfit the training data, while the number of steps required for generalization increases rapidly. When $\alpha \leq 30\%$, the model would not generalize in 10^5 steps.

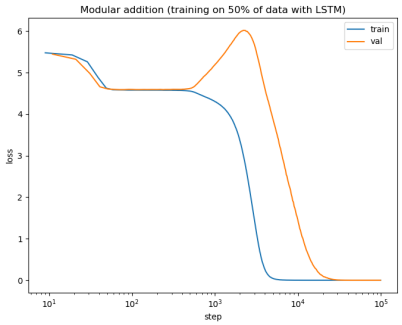
3.2 Grokking Phenomenon of Other Models

We study the grokking phenomenon for two other network architectures, long short-term memory (LSTM) and multilayer perceptron (MLP), suggesting that grokking is not unique for transformer. Specifically, our LSTM has 2 layers, a hidden state of size $d_{\text{hidden}} = 128$, and dropout 0.1. Our MLP has 2 hidden layers of size $d^{(1)}_{\text{hidden}} = d^{(2)}_{\text{hidden}} = 256$, both of which have dropout 0.1. Other hyperparameters remain the same as Section 3.1.

The grokking curves for LSTM and MLP are shown in Figure 3 and Figure 4, respectively.



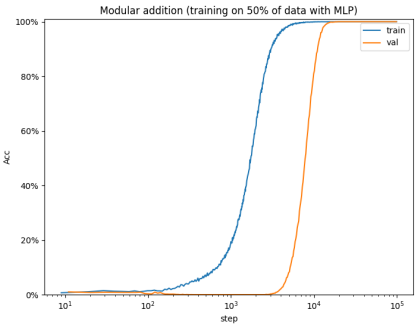
(a) Training and validation accuracy



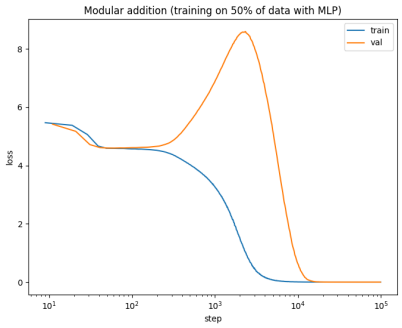
(b) Training and validation loss

Figure 3: The grokking curves of LSTM model

In Figures 3a and 4a, we again observe a delay of generalization which, however, is not as significant as that in Figure 1a. Besides, a similar increase of validation loss could be observed in both Figure 3b and Figure 4b.



(a) Training and validation accuracy



(b) Training and validation loss

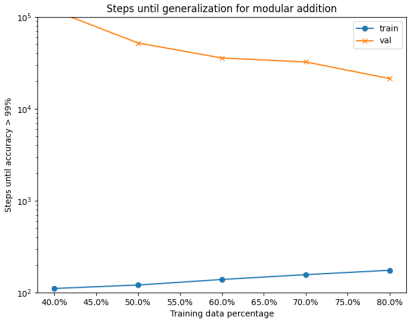
Figure 4: The grokking curves of MLP model

3.3 Effects of Different Hyper-Parameters

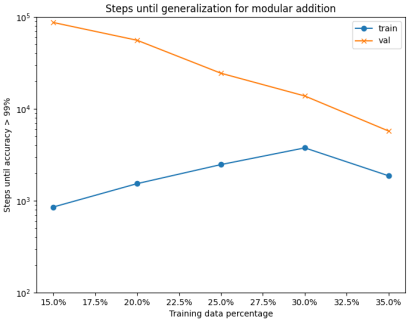
3.4 Grokking for K-Wise Modular Addition

In this task, we investigate the grokking phenomenon of K-wise modular addition. Due to the limitation of GPU memory, we only perform experiments on $2 \leq K \leq 5$, and $p = 31$. Specifically, we attempt to discover grokking phenomenon by adjusting training data fraction α, and use techniques such as weight decay and drop out to lower down the scale of training data as much as possible. We use the model Transformer which has the default parameters as shown in 3.1.

The minimal training steps for memorization and generalization on different alpha when $K = 2, 3$ is plotted in Figure 5. Comparing Figures 5a and 5b,we discover that when $K = 3$, the grokking could happen at lower α than when $K = 2$, but the gap between memorization and generalization is not as obvious as when $K = 2$.



(a) Grokking phenomenon for $K = 2$



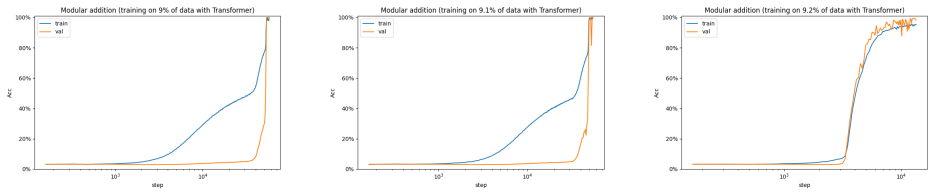
(b) Grokking phenomenon for $K = 3$

Figure 5: Grokking phenomenon for different K and α

This above conclusion is also applied for $K = 4$, but the grokking phenomenon is harder to discover. Figure 6 illustrates the behaviors for $K = 4$ with small variance α. Figures 6a and 6b show that grokking phenomenon would happen before the training accuracy reaches 60%, and Figure 6c shows that it is also possible that grokking would not happen with a small disturbance to α. Besides, we don't find grokking phenomenon within 10^5 steps when $\alpha \leq 8\%$.

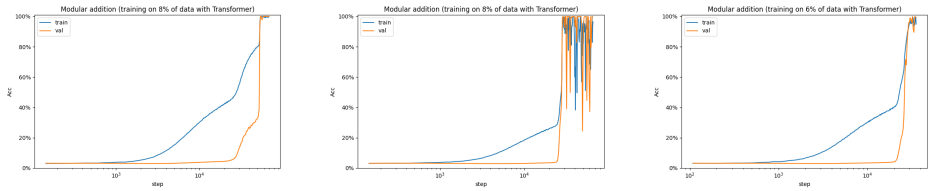
In summary, the experiments imply that the model has better generalization performance when K is larger, with smaller training data fraction, but the grokking phenomenon is less apparent.

In addition, as Figure 7 illustrates, we find that with proper adjustment to weight decay and dropout, the model could generalize with fewer training data, but sacrifice training stability.



(a) $\alpha = 9.0\%$ (b) $\alpha = 9.1\%$ (c) $\alpha = 9.2\%$

Figure 6: Grokking phenomenon for $K = 4$ with small variance α



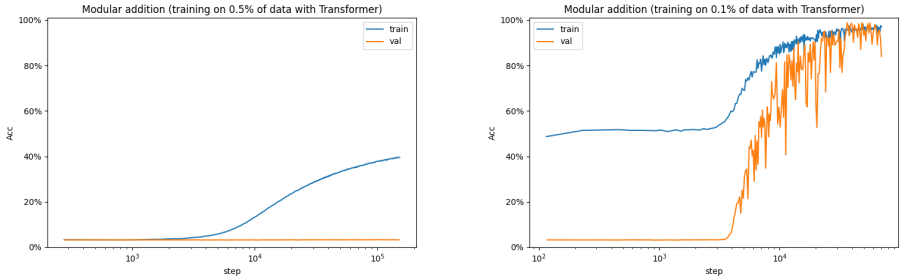
(a) $\alpha = 8.0\%$ weight decay = 0.1 dropout = 0.2

(b) $\alpha = 8.0\%$ weight decay = 0.2 dropout = 0.3

(c) $\alpha = 6.0\%$ weight decay = 0.2 dropout = 0.3

Figure 7: Grokking phenomenon for $K = 4$ with different weight decay and dropout

We further investigate the situation when $K = 5$. Since it has an amount of more than 2×10^7 data, we could only let $\alpha \leq 0.5\%$[2]. This greatly limits our model's generalization ability and Figure 8a suggests no sign of grokking within 10^5 training steps. Then we try to modify training dataset by adding all equations of $K = 2, 3$ to it. Surprisingly the model successfully generalizes within 10^5 steps with no more than 0.1% training data fraction (not include equations we added) as Figure 8b shows.[3] This suggests that the modification to training set may be effective. We look forward to further study about this technique on the larger K.



(a) Train and val accuracy of $\alpha = 0.5\%$ without modified data

(b) Train and val accuracy of $\alpha = 0.1\%$ with modified data

Figure 8: Grokking phenomenon for $K = 5$

4 An Explanation of the Grokking Phenomenon

In this section, we provide an explanation of the grokking phenomenon based on [2], which claims that grokking happens as a transition between different regimes of training. We first briefly review

[2]Validation dataset is also limited. The dataset has been split into a training set and a validation set in a ratio of 1:3.

[3]The training accuracy quickly raises to about 50% because the added equations account about 50% in the training set. It seems that the model quickly learns the case of $K = 2, 3$.

the kernel regime and rich regime defined in [2] in Section 4.1, and illustrate how they help explain the grokking phenomenon for modular addition in Sec XXX.

4.1 Kernel Regime And Rich Regime

We have the following definition of neural tangent kernel (NTK).

Definition 4.1 (Neural Tangent Kernel [3, 2])**.** Let Θ be the parameter space and $\mathcal{X}$ be the input space. Let $f\colon \Theta \times \mathcal{X} \to \mathcal{Y}$ be a neural network. For $\theta \in \Theta$, the *neural tangent kernel* of $f(\theta, \cdot)$ is defined as

$$K_\theta(\mathbf{x}, \mathbf{x}') := \nabla_\theta f(\theta, \mathbf{x}) \nabla_\theta f(\theta, \mathbf{x}')^\top$$

References

[1] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *CoRR*, abs/2201.02177, 2022.

[2] Tanishq Kumar, Blake Bordelon, Samuel J. Gershman, and Cengiz Pehlevan. Grokking as the transition from lazy to rich training dynamics. In *ICLR*. OpenReview.net, 2024.

[3] Arthur Jacot, Clément Hongler, and Franck Gabriel. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, pages 8580–8589, 2018.

A Appendix

Optionally include extra information (complete proofs, additional experiments and plots) in the appendix. This section will often be part of the supplemental material.