Group 24: An Analysis of Task Learning Capabilities of Grokking in Neural Network Models

Jasper Nie (20285349)

Vincent Hung (20273117)

Brandon Cheng (20269419)

CISC/CMPE 452

Neural and Genetic Computing

December 2, 2024

Contents

1 Motivation

With the rise of AI and large language models such as ChatGPT, the understandability of these extremely complicated deep learning models poses an important issue to address: we do not know how the model makes its decisions due to the high number of hidden layers and neurons within the neural network. So for example, when ChatGPT produces hallucinations and or wrong information, there is no way for developers to determine what is bad and the only solution is to continue training the model and praying that the hallucination is fixed with enough brute-force examples.

A deep learning model will find an extremely complicated solution to whichever problem it is designed to model, but a relatively shallow neural network with the same accuracy can give a more explainable solution, something that developers and users of the neural network can understand.

The problem with shallow neural networks is that they suffer from a couple of issues that affect their performance. The most notable problem is that they tend to overfit, meaning they will opt to memorize the training set instead of learning how to generalize and produce correct answers for inputs not in the training set. This is where the phenomenon of grokking can potentially be useful for a solution to both of these problems. This is when under certain parameters, a shallow neural network can learn to generalize way past originally overfitting to the training set. More will be discussed in the problem description section. The goal of this paper is to find out which parameters specifically contribute to this phenomenon and what kind of dataset can it be applied to.

2 Problem Description

The phenomenon of grokking has been a relatively new discovery. As shown in the original paper in 2021[1], grokking was first discovered when training on synthetic datasets like modular addition. Our project aims to investigate how grokking can be applied to more complex tasks with noisier data. Specifically, we chose tabular data classification and image classification as the next logical steps to induce grokking. We believe that by understanding which tasks can "grok" and which can not, we can valuable information about the grokking phenomenon and we could encourage its appearance in situations such as a lack of data where grokking may be the only viable path to generalization

3 Contribution

3.1 Brandon

Literature Review and Experiment Design: Brandon conducted the literature review and designed the experiments for our project. They reviewed relevant papers and researched together insights and theories that could apply to our project. Additionally they designed the experimental framework. Their work was crucial to ensuring that our experiments were well-informed and methodologically sound.

3.2 Vincent

Coding: Vincent took on the majority of the coding work for the project. They were responsible for training code for both wine classification and MNIST image classification. Vincent created the code using PyTorch, which was used for training in the experiments. Their work provided the initial framework and helped to train and test our models.

3.3 Jasper

Fine tuning and experimentation execution: Jasper was responsible for the majority of the training in the group. They ran multiple different training runs of the models. Experimented with various hyperparameters and loss functions. Jasper's work was crucial in finetuning the models and ensuring we explored a wide range of configurations while exploring grokking.

4 Related Work

4.1 Paper 1: Grokking: Generalization Beyond Overfitting On Small Algorithmic Datasets

[1] This paper first defined and studied the grokking phenomenon. Grokking is a rather new but interesting occurrence where a neural network can be seen to generalize even after over-fitting the training data. The paper highlighted the conditions this phenomenon was discovered, providing an analysis of the conditions that were conducive to this phenomenon. It identified key factors that were believed to encourage grokking, with weight decay being a key element. It is believed that weight decay helps in regularizing the model by penalizing large weight values, thus simplifying and generalizing the model.

However, this paper is limited to synthetic datasets, specifically modular addition. This does not represent the complex real-world tasks we assign neural networks to complete. Our work aimed to extend the research done on grokking by exploring grokking in more complex tasks such as tabular data classification and optical character recognition. By doing so we aim to better understand the applicability of grokking on tasks beyond synthetic data. This exploration will help clarify what tasks or data can be grokked or not, providing insight into the generalization capabilities of neural networks.

4.2 Paper 2: PROGRESS MEASURES FOR GROKKING VIA MECHANISTIC INTERPRETABILITY

[2] This paper further explored grokking by reverse engineering the circuits in models pre and post-grokking. It developed progress measures for the grokking process and provided important insights into why weight decay is crucial. The visualization of the internal circuits pre and post-grokking presented in this paper demonstrated the emergence of the phenomenon in an easy-to-understand method. While this paper focuses primarily on understanding the internal mechanism of grokking, our work builds on these insights by applying the grokking process to different types of data, aiming to identify the limits and potential of grokking in real-world applications.

4.3 Paper 3: OMNIGROK: GROKKING BEYOND ALGORITHMIC DATA

[6] The Omnigrok paper aims to tackle two different questions. "Why is generalization much delayed after over-fitting?" and "can grokking occur on datasets other than algorithmic datasets?" we will focus on the latter as it is the most relevant to the goal of this project. The paper demonstrates and concludes that grokking can occur for a variety of tasks, including sentiment analysis, molecule property prediction and, most importantly, image classification. Although less than algorithmic datasets, the authors observed grokking signals that have been attributed to representation learning.

5 Datasets Used

For this paper, we decided to two 3 datasets. The first dataset is the synthetic dataset used in the original paper introducing the grokking phenomenon. The paper uses a small algorithmic database for modular arithmetic tasks such as modular addition and modular multiplication. These datasets consist of input-output pairs, where the inputs are integers and the results are under a specific modular operation for a given modulus p. For this speicifc dataset, we just ran the code repository of the original paper[1] to verify the grokking phenomenon.

The second dataset is a widely-used classification dataset called the Win Quality Dataset[4]. This dataset contains chemical and sensory data for red and white wines for Vinho Verda in Portugal. There are 11 different chemical properties and then each wine sample has a quality score rated from 0-10.

The last dataset is an image classification dataset called the MNIST dataset[5]. It contains over 70,000 grayscale images of single-digit integers from 0-9 in a 28x28 pixel resolution. Each pixel has a value between 0 (black) and 255 (white) and these values are normalized to between 0 and 1 in preprocessing.

6 Implementation

For this section, the group worked together on a shared model for MNIST[5] and the quality dataset. Thus there is one shared implementation for the model implementation.

6.1 Data Preprocessing

For our experiments, we utilized MNIST[5] and winequality datasets[4]. The reprocessing steps were minimal and were completed collectively. The standard train-test split was applied to both datasets. Some early experimentation was done on the proportion of the full dataset being utilized. Specifically we experimented with very small portions of data – 20-40 percent – before deciding to use the entire dataset. This helped us check claims of small dataset sizes being more grokking friendly.

6.2 Experimental Setup

The experimental setup implementation was mainly completed by Vincent, with Brandon providing the architecture and initial parameters for each model. Since the initial paper we chose utilized PyTorch, we continued with this framework instead of TensorFlow for neural network training.

6.3 Training

Although the training was a collaborative effort, the majority of the training relied on Jasper. He was responsible for overseeing the training process and conducting most of the model training.

6.4 Testing

Due to the nature of the experiments, instead of evaluating testing accuracy, we look for a sudden increase in validation accuracy after the point of over-fitting to determine the presence of grokking. This approach allows us to identify even slight grokking as we can see if validation accuracy trended upwards while the training accuracy was over-fitting.

6.5 Validation

The validation accuracy was calculated based on a separate test set and plotted along with the training accuracy to be used in the identification of grokking. This was used to visually identify the grokking phenomenon.

7 Results and Discussion

7.1 Results

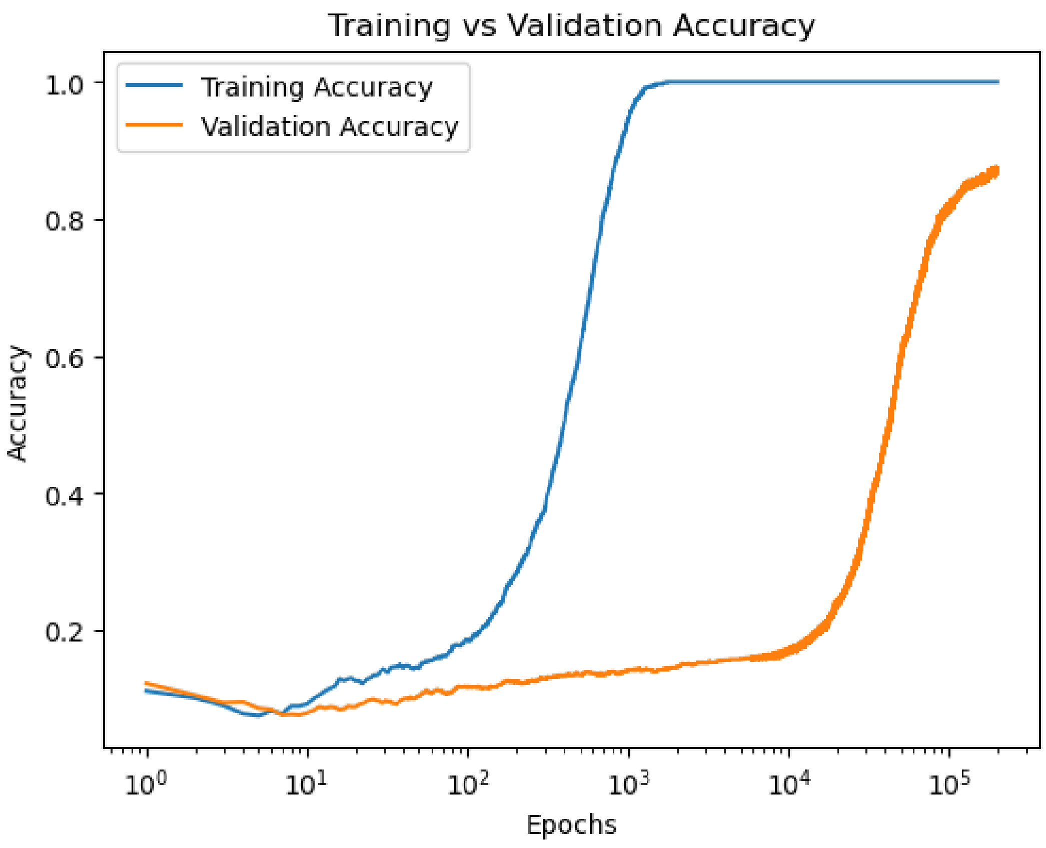

Figure 1: Training Results with Weight Normalization on MNIST Dataset[5]

Epochs	Train Accuracy	Validation Accuracy	Weight Normalization
20000	1.0000	0.2370	125.3902
40000	1.0000	0.4750	101.8117
60000	1.0000	0.6700	84.4514
80000	1.0000	0.7760	72.9903
100000	1.0000	0.8140	65.5316
120000	1.0000	0.8390	60.8339
140000	1.0000	0.8520	57.4474
160000	1.0000	0.8620	55.7822
180000	1.0000	0.8640	53.2813
200000	1.0000	0.8720	51.6976

Table 1: Training Results with Weight Normalization on MNIST Dataset

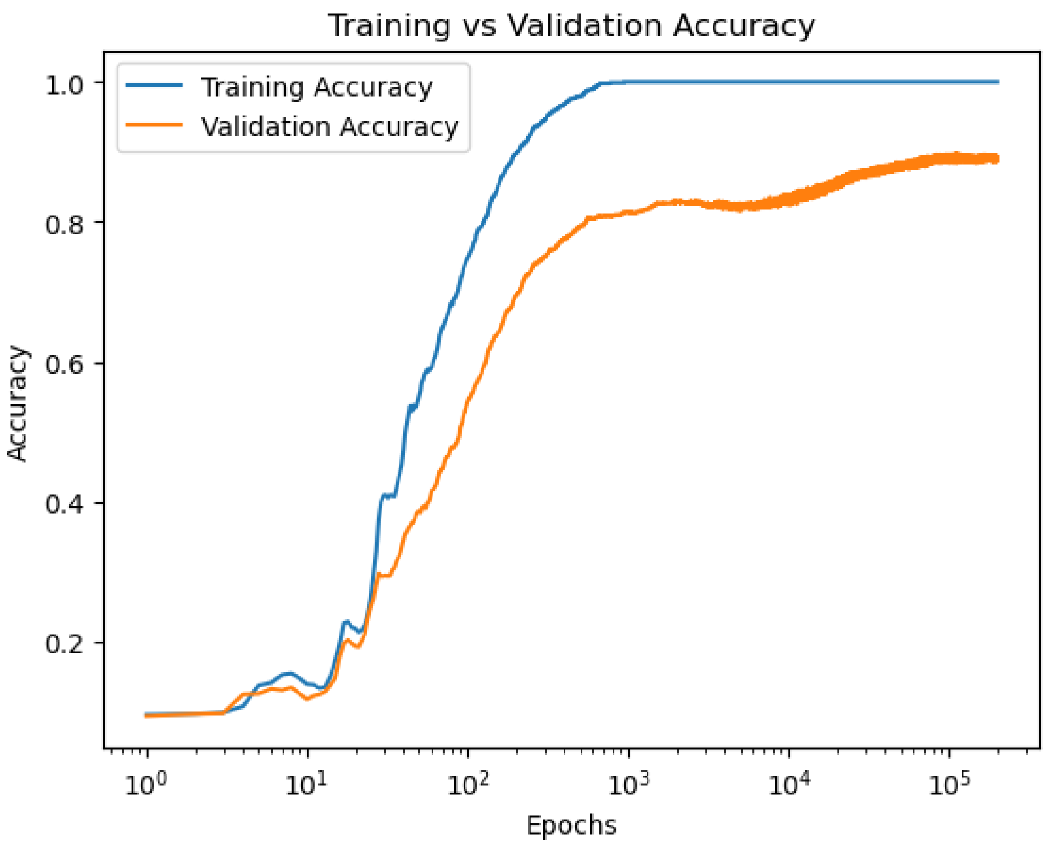

Figure 2: Training Results with Weight Normalization on MNIST Dataset[5] (Sigmoid Activation Function Hidden Layers)

Epochs	Train Accuracy	Validation Accuracy	Weight Normalization
20000	1.0000	0.8560	141.8049
40000	1.0000	0.8780	132.7365
60000	1.0000	0.8820	128.6327
80000	1.0000	0.8890	126.4279
100000	1.0000	0.8930	124.6334
120000	1.0000	0.8910	122.8214
140000	1.0000	0.8890	120.7898
160000	1.0000	0.8890	118.7557
180000	1.0000	0.8910	116.6412
200000	1.0000	0.8910	114.8501

Table 2: Training Results with Weight Normalization on MNIST Dataset (Sigmoid Activation Function Hidden Layers)

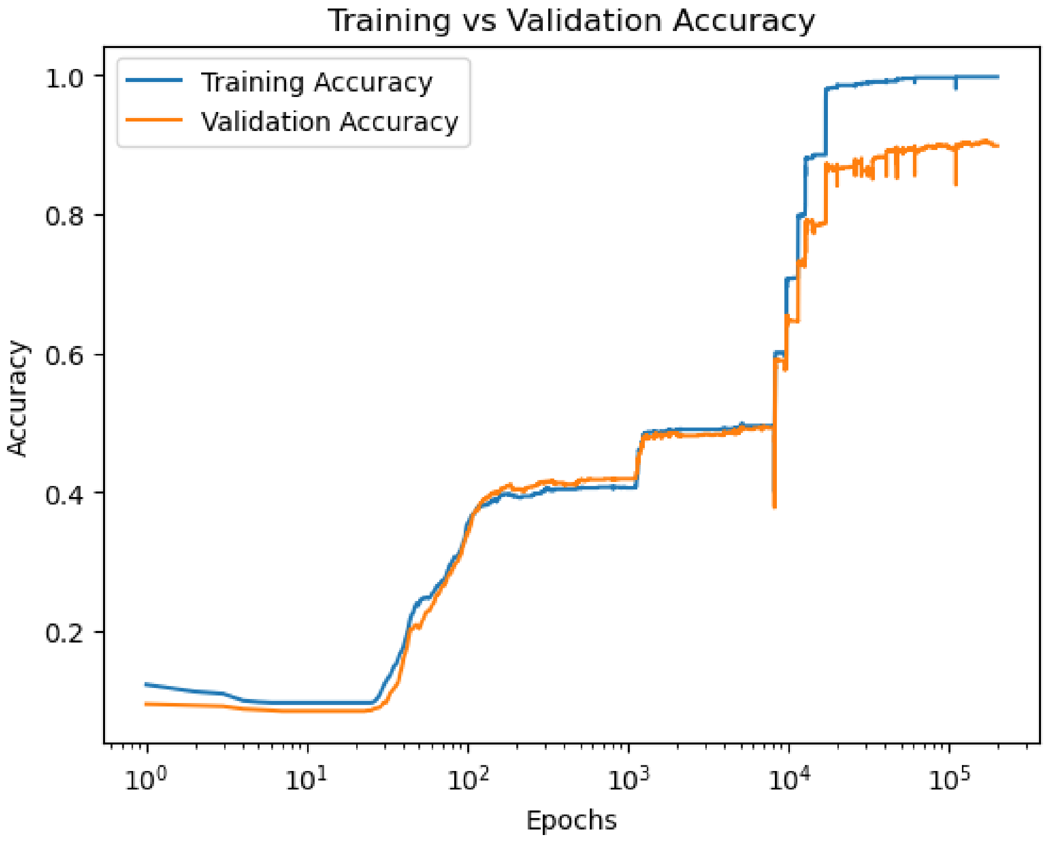

Figure 3: Training Results with Weight Normalization on MNIST Dataset[5] (Sigmoid Activation Function Output)

Epochs	Train Accuracy	Validation Accuracy	Weight Normalization
20000	0.9820	0.8650	132.0966
40000	0.9900	0.8820	111.8685
60000	0.9950	0.8920	95.8474
80000	0.9960	0.8960	81.4139
100000	0.9960	0.8960	68.4259
120000	0.9970	0.8990	61.8850
140000	0.9970	0.9000	53.1198
160000	0.9970	0.9040	46.6494
180000	0.9970	0.9020	41.8671
200000	0.9970	0.8980	38.3079

Table 3: Training Results with Weight Normalization on MNIST Dataset (Sigmoid Activation Function Output)

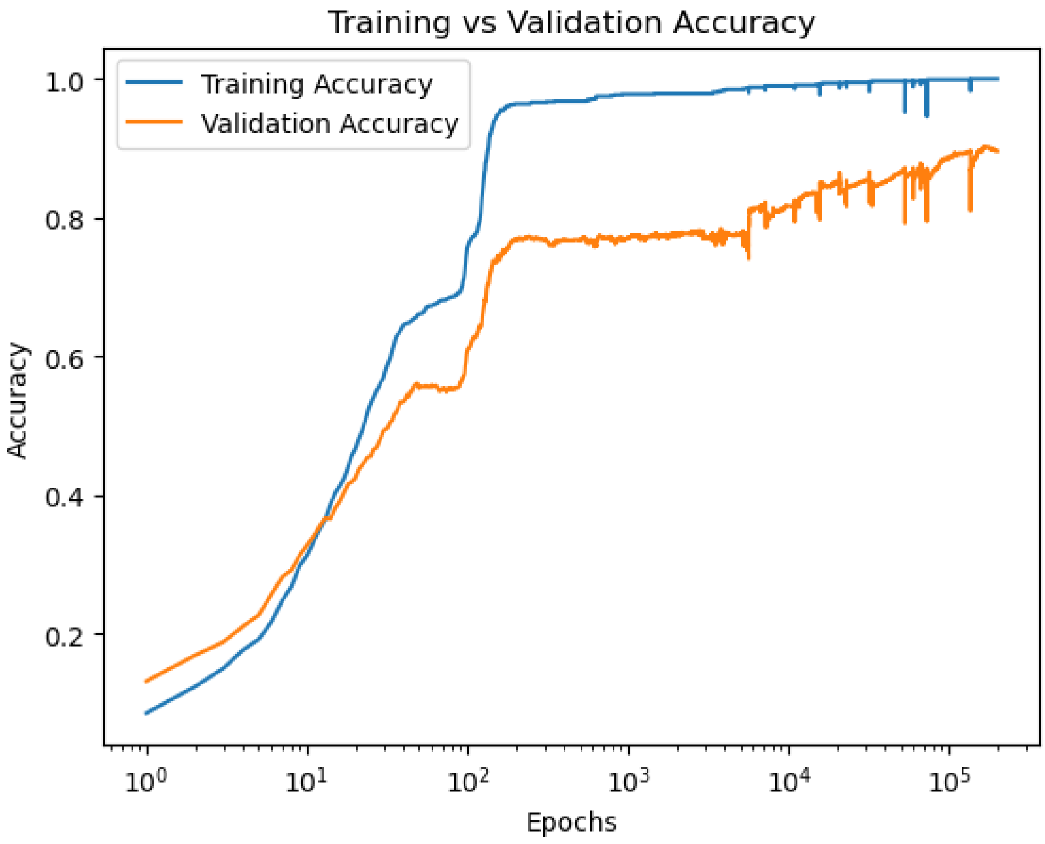

Figure 4: Training Results with Weight Normalization on MNIST Dataset[5] (Softmax)

Epochs	Train Accuracy	Validation Accuracy	Weight Normalization
20000	0.9940	0.8510	128.8513
40000	0.9970	0.8510	110.1419
60000	0.9970	0.8560	94.1113
80000	0.9990	0.8680	81.6418
100000	0.9990	0.8880	67.5138
120000	0.9990	0.8910	56.6119
140000	1.0000	0.8770	52.2332
160000	1.0000	0.8970	44.4891
180000	1.0000	0.9000	38.6628
200000	1.0000	0.8960	34.3330

Table 4: Training Results with Weight Normalization on MNIST Dataset (Softmax)

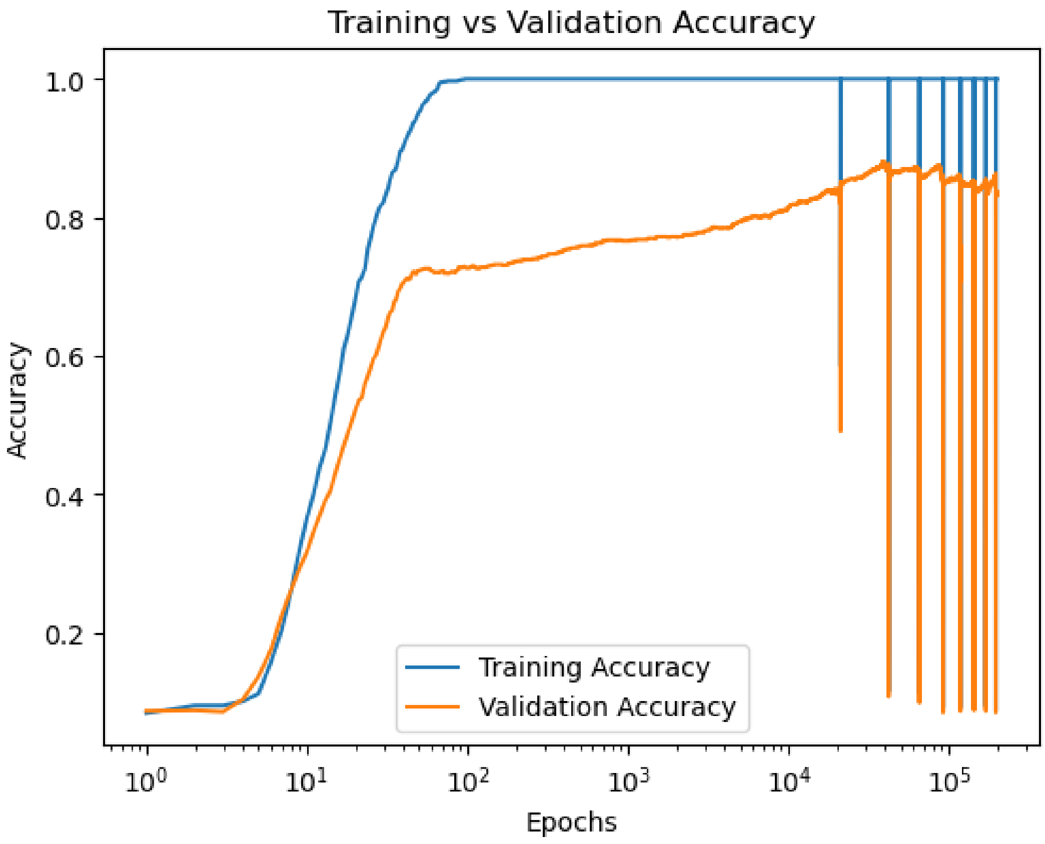

Figure 5: Training Results with Weight Normalization on MNIST Dataset[5] (Cross Entropy)

Epochs	Train Accuracy	Validation Accuracy	Weight Normalization
20000	1.0000	0.8370	129.9063
40000	1.0000	0.8750	113.9737
60000	1.0000	0.8720	104.5435
80000	1.0000	0.8690	106.2238
100000	1.0000	0.8540	127.8685
120000	1.0000	0.8490	132.6842
140000	1.0000	0.8500	124.5467
160000	1.0000	0.8460	144.3296
180000	1.0000	0.8460	171.7871
200000	1.0000	0.8360	218.4787

Table 5: Training Results with Weight Normalization on MNIST Dataset (Cross Entropy)

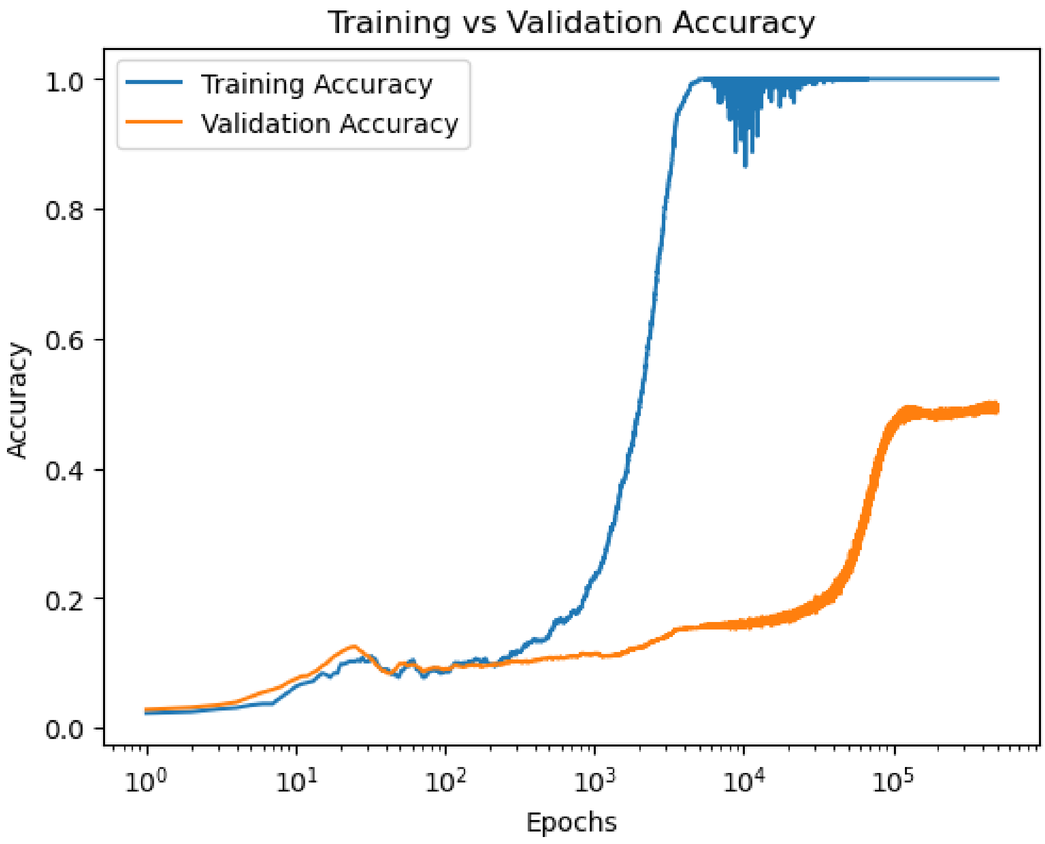

Figure 6: Training Results with Weight Normalization on Wine Quality Dataset[2]

Epochs	Train Accuracy	Validation Accuracy	Weight Normalization
50000	1.0000	0.2472	99.1951
100000	1.0000	0.4711	64.8165
150000	1.0000	0.4840	55.0575
200000	1.0000	0.4792	52.8173
250000	1.0000	0.4847	52.2982
300000	1.0000	0.4872	52.0968
350000	1.0000	0.4858	51.9308
400000	1.0000	0.4924	51.8967
450000	1.0000	0.4926	51.8224
500000	1.0000	0.4929	51.3104

Table 6: Training Results with Weight Normalization on Wine Quality Dataset

7.2 Discussion

Our experiments with the MNIST[5] (Cross Entropy) dataset suggest that the observed grokking phenomena may not actually indicate that the MNIST[5] (Cross Entropy) classification task is experience grokking. Instead, it appears that using a linear output layer and mean squared error (MSE) loss introduces an algorithmic component to the overall task and it is this element that is being ”grokked” instead of the actual classification. This is shown in the results above, where all of the MNIST[5] (Cross Entropy) models, regardless of grokking, converge to 85-90% accuracy. Furthermore, models that do exhibit grokking (see Figure 1 and Table 1) tends to perform much worse at the beginning compared to non-grokking models, suggesting that the setup used to achieve grokking, rather than enabling better generalization, may merely obstructs it at the beginning, resulting in a delayed generalization curve.

Our findings suggest that the results from the Omnigrok[6] (Cross Entropy) paper may not actually prove that grokking can occur in non-algorithmic tasks.

8 Conclusion and Future Work

We conclude that rather than certain circumstances allowing grokking to occur in non-algorithmic tasks, as claimed in Omnigrok[6], it is possible that certain circumstances instead artificially create algorithmic tasks within a model that blocks the learning of the non-algorithmic task, forcing the model to go through a grokking process to resolve the algorithmic task before it can learn to generalize to the original non-algorithmic task. We further suggest that this explains the presence of grokking on the MNIST[5] data as documented by the Omnigrok[6] paper.

Thus, through our conclusion, we contribute to ongoing research on grokking by identifying a possible source of error for current and future endeavors aiming to prove or disprove the "grokkability" of non-algorithmic tasks.

While we have challenged the conclusions of past works regarding the presence of grokking in non-algorithmic tasks and provided a theoretical explanation for why we think their findings are flawed, we have not yet:

- Validated our theory - We suggest isolating the circumstances that cause grokking in non-algorithmic tasks, and seeing if grokking emerges even when those circumstances are applied to trivial tasks.
- Proven or disproven the presence of grokking in non-algorithmic tasks - We suggest further research in this field explore ways to purge possible algorithmic tasks from emerging within non-algorithmic tasks

Thus, we leave these topics open as promising avenues for future research, inviting further exploration and investigation to build upon the findings presented here.

9 References

[1] Power, A., Burda, Y., Edwards, H., Babuschkin, I., & Misra, V. (2021). Grokking: Generalization beyond overfit. Grokking: Generalization Beyond Overfitting On Small Algorithmic Datasets. https://mathai-iclr.github.io/papers/papers/MATHAI_29_paper.pdf

[2] Nanda, N., Chan, L., Lieberum, T., Smith, J., & Steinhardt, J. (2023). Progress Measures For Grokking via Mechanistic Interpretability. http://arxiv.org/pdf/2301.05217

[3] Power, A., Burda, Y., Edwards, H., Babuschkin, I., & Misra, V. (2022). Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets. arXiv.org. https://arxiv.org/abs/2201.02177

[4] Aeberhard, S., & Forina, M. (1994). Wine. UCI Machine Learning Repository. https://archive.ics.uci.edu/dataset/109/wine

[5] LeCun, Y., Cortes, C., & Burges, C. J. C. (1994). The MNIST database. MNIST Handwritten Digit Database. https://yann.lecun.com/exdb/mnist/

[6] Liu, Z., Michaud, E., & Tegmark, M. (n.d.). Grokking beyond algorithmic data. OMNIGROK: GROKKING BEYOND ALGORITHMIC DATA.https://arxiv.org/pdf/2210.01117