Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Investigating "Grokking" – Late Generalisation in Deep Learning

Bachelor's Thesis

Benjamin Arn

barn@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Peter Belcák
Prof. Dr. Roger Wattenhofer

March 22, 2023

Abstract

"Grokking" is the phenomenon of late generalization, where machine learning models jump to perfect accuracy after being in the overfitting regime for a long time. Lacking any obvious explanation, understanding this phenomenon may lead to new insights for modern machine learning theory. Grokking was discovered fairly recently in 2022 by Powers et al. [1], and in this thesis, I replicated their experiments to get a more in-depth understanding of this phenomenon. My experiments and analysis uncovered irregular training loss spikes as a fundamental part of grokking behaviour and I describe the details of their occurrence. In addition, I point out different irregular learning behaviours that suggest to look at grokking as an intricate, context-sensitive behaviour instead of a well-defined incidence.

Contents

Chapter 1

Introduction

In machine learning, overfitting is a well known problem in theory as well as in application: a machine learning model can learn training data too "well" to generalize to any unseen input, either by training the model for too long or because the model is too complex. A common solution to this problem is early stopping: the learning process is stopped when generalization capabilities of a model do not improve for a certain amount of time. This is justified by assuming the generalization capability curve of a model to be in a U-shape: Descending at first while learning relevant structure of the training data, then ascending again after fitting too close to it.

This proves to be an oversimplification, as the discovery of "grokking" [1] by Powers et al. in 2022 shows. In this phenomenon, the generalization ability of a model improves suddenly from guessing randomly to perfect generalization, a long time after severely overfitting to the training data. Like double descents [2, 3], this behaviour challenges classical machine learning paradigms. This is especially interesting since this grokking behaviour happened using a transformer model, which is a state-of-the-art architecture for sequential inputs and responsible for recent advancements in natural language processing. Additionally, related work [4, 5] seems to agree that grokking strongly correlates to representation learning, which is the process of finding relevant structures in complicated data like pictures, videos and text and thus essential to many applied machine learning tasks.

Understanding of this late generalization called grokking might lead to more general understanding about these complicated models, and the uncertainty about its causes indicates a limit in current machine learning theory. To document this phenomenon, I will stick closely to the initial work by Powers et al. [1] and try to reproduce their findings. This thesis is about the data gathered with my experiments, with which I will present a more comprehensive picture of grokking that also shows its vagueness and ambiguity.

Chapter 2

Related Work

The work of Powers et al. laid the foundation for this thesis in January 2022 in describing late generalization for algorithmically created datasets for the first time and calling it "grokking" [1]. The authors have written the code that I have used in my experiments. Besides describing and visualizing grokking, they draw the connection between dataset size and generalization ability of their transformer model and investigate the impact of different optimization techniques on data efficiency.

In the short time since then, several other researchers investigated this strange phenomenon. In the paper Liu et al. released in October [4], the authors break down grokking behaviour as a problem of representation learning. With framing grokking as part of the interplay between a encoder and decoder, they introduce different phases of learning behaviour with the introduction of a simpler toy model while only considering modular addition.

A few weeks after, a subset of the same authors released another paper [6] in which they explain grokking behaviour by looking at loss landscapes and analizing model weight size, again with the toy setting they introduced before [4]. By strongly regulating model weights, they achieve the introduction of grokking to other, more common datasets like MNIST as well as the removal of grokking from algorithmically created datasets. They highlight the especially high dependance of algorithmically created datasets to learn good representations.

Very similar to what I observed in my experiments about spikes in training loss and their importance on the grokking behaviour, Thilak et al. stated in [7] that optimization anomalies coming from adaptive optimizers very late in the learning process may be essential for grokking. They identify cyclic phases of stable and unstable training regimes related to the norm of the last layer weights and term it the "slingshot mechanism".

Early in 2023, Gromov managed in [5] to achieve grokking on just a fully-connected two-layer network on modular arithmetic tasks (a subset of the algorithmically created dataset used in [1]) and showed that no regularization and nothing other than vanilla gradient descent is necessary for grokking behaviour.

On one hand, this shows grokking as not dependent on a complex architecture or optimizer, and on the other hand makes the learned representation humanly analyzable by having just two layers of weights. Gromov's work emphasizes the notion of the work of Liu et al. [4] to look at grokking as tightly interconnected to the representations that have to be learned by the model to solve the task, but it challenges the notion that a clear encoder-decoder dichotomy is responsible for grokking.

In the most recent work regarding grokking [8], Davies et al. lay the theoretical groundwork to understand grokking and double descents as different effects of a single learning dynamic. They introduce a mathematical model with which the learning behaviours of both different phenomena can be produced, connecting both learning behaviours that go against classical machine learning intuition.

Chapter 3

Experiment

My experiments were modeled to replicate the findings of the paper by Powers et al released in 2022 [1]. I used the code the authors submitted on Github with as few modifications as possible. The code implemented a 2-layer transformer model based on the pytorch lightning module for python, using an AdamW optimizer.

The model is trained to predict solutions of relatively simple equations. For example: $x + y$ modulo 97 for all numbers from 0 to 96, leading to a dataset of 9409 equations. To add an abstraction layer, the numbers are substituted by characters, so that the equations are written in the form $a * b = c$. As the model has no information on the operation that generated the dataset, it effectively has to learn it's own representation of the operation.

There where different operations of varying complexity implemented in the code. I limited myself to few of them to be able to make accurate observations. To enhance the generality of the experiments, I also used datasets derived from the multiplication tables of arbitrary groups with the same size. Thankfully, I could use the group datasets my fellow student Simon Peter created in his Bachelor's Thesis [9].

Operations used for which the model achieved generalization:

- Addition and Subtraction (Modulo 97)
- Multiplication and Division (Modulo 97)
- $x^2 + y^2$ (Modulo 97)
- 3 different groups from Simon Peter's created dataset of the same size

Operations used for which the model did not achieve generalization:

- Division for even numbers, subtraction for odd numbers (Modulo 97)
- $x^3 + x * y$ (Modulo 97)

I focused on comparing the different datasets with varying train-validation-split percentages, with double the amount of optimization steps used compared to the original experiment by Powers et al. [1]. Note that I did not vary other hyperparameters used between different datasets with the objective of better comparability. I included experiments without generalization in my analysis as there is still interesting behaviour to be shown. A more detailed look into the specific process of reproducing this experiment can be found in Appendix A.

Chapter 4

Results

4.1 Grokking is Reproducible

The late generalization behaviour called "grokking" has occurred for datasets generated from several different operations, including the dataset generated on my own from multiplication tables of arbitrary abstract groups. As seen in Figure 4.1 from my own experiments, two features are very pronounced in the ideal version of grokking as described in the paper by Powers et al. (2022) [1]:

- Perfect validation accuracy is achieved a long time after fitting to the training data (up to a 100 times later)
- The validation accuracy jumps in few steps from extremely low to perfect accuracy

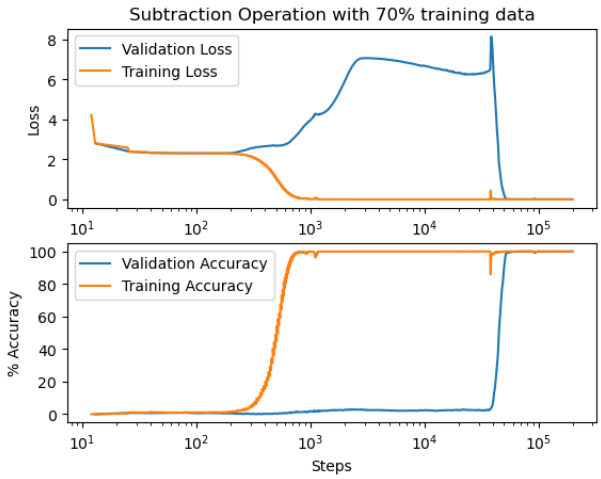


Figure 4.1: A model case of grokking

In my experiments, this kind of grokking behaviour did happen, but only seldom as explicit as in this example. While late jumps in validation accuracy

happened regularly, the jumps varied in size and exact location and were sometimes preceded with a slow, continuous rise in accuracy, discussed later in Section 4.5.

4.2 Relating grokking to Training Dataset Size

An interesting influence on the generalization behaviour of the model are the train-validation-splits. In machine learning, more data usually increase the chance that the model gets generalization capacity. Grokking seems to be located in a space where the model is theoretically capable of generalizing, but not able to make steady, slow improvements to reach that point.

To analyze the grokking phenomenon, the authors Liu et al. released their paper "Towards Understanding Grokking: An Effective Theory of Representation Learning"[4]. Inside, they split up four phases of learning performance by introducing new terminology: confusion, where the model cannot even be fitted to the training data; memorization, where the model overfits to the training data without any generalization capability; comprehension, where a certain level of generalization is achieved while fitting to the training data; and lastly grokking, which is said to be a phase happening in between memorization and comprehension.

This approach gives a good intuition for the model's learning behaviour, especially the placement of grokking in between memorization and comprehension, which has shown to be accurate relating to different sizes of training datasets. In Figure 4.2, the three phases of memorization, grokking and comprehension can be seen in a relatively small range of train-validation-splits.

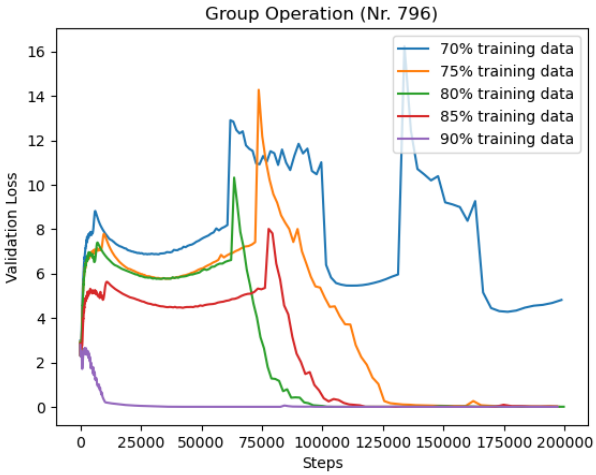


Figure 4.2: Relating validation loss to training dataset size as a fraction of the whole dataset

However, the approach of having different phases implies their clean separability which cannot be concluded from my findings. Sometimes, there are jumps in validation accuracy that are still not sufficient to reach real generalization capabilities of the model, signifying something between memorization and grokking. Another time, there is a slow rise in validation accuracy which is capped off at the end with a grokking-like spike, resulting in perfect generalization, signifying something between grokking and comprehension. These special behaviours will be addressed later in section 4.5.

My experiments highlight a clear relation between the amount of data and the learning performance of the model. In all my experiments, if fast generalization occurred, grokking occurred too when feeding less data from the same dataset to the model. Conversely, if grokking occurred, fast generalization occurred too with more data from the same dataset.

4.3 Timing of Grokking

It should be obvious that measurements on the validation data is nothing that the model is aware of, but a way for us observers to look at the model's capabilities. By defining grokking as a spike in validation accuracy, we are at risk of ignoring causality: changes in the model's weights should be reflected in its training loss, since that is what every optimizer tries to minimize. It seems that the training loss does not really change after a short period of fitting to the training data when we look at figure 4.1 from earlier.

However, this is only true when looking at absolute loss values. When we change to logarithmic plots (figure 4.3), we can see that our training loss gets very close to zero, but spikes again after some time by a factor of over 10^6, inducing our sought-after grokking behaviour.

The learning behaviour of the model is tightly coupled with the training loss. As a consequence, the model will not change if the training loss is sufficiently low and cannot be decreased with small changes in the model weights, commonly known in machine learning as a local minimum. This could explain the extreme time delay of grokking: The model might be floating in a local minimum with no escape, until the loss spikes again and enables the model to change for better. Although, this does not imply that the model changes for better when a training loss spike occurs, as there can be multiple training loss spikes without the model getting generalization capabilities. Both cases can be seen in figure 4.4. These cases show that the spikes of the validation and training losses mirror each other, with each spike generating an opportunity for the curves to get closer to each other.

It should be noted how much the timing of the late generalization varies. I have not seen an upper bound where I can be certain that the model does not get

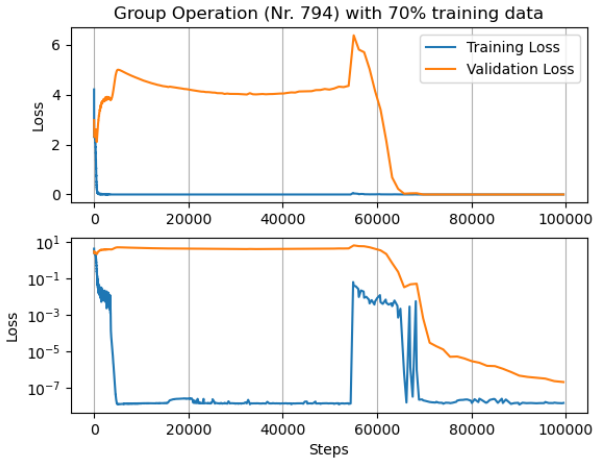


Figure 4.3: Comparing absolute values to a logarithmic scale

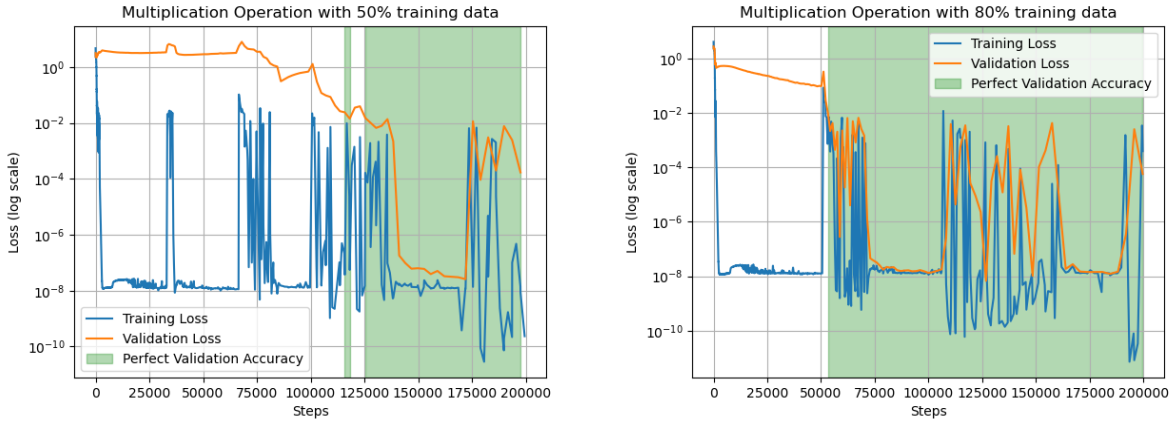


Figure 4.4: Two Examples on the relation between Training Loss and Validation Loss

generalization capabilities. I doubled the range from the 10^6 optimization steps in the original paper by Powers et al. (2022) to $2 * 10^6$, whereby the jump to full accuracy in figure 4.4 even can be seen.

On the other hand, the lower bound for grokking is the first training loss spike. If the model can not generalize after fitting to the training data in the first 100 steps, generalization will not happen until the training loss gets disturbed enough for the model to find a new, better minimum. From looking at these graphs alone, it hard to determine if these training loss spikes and subsequent grokking are an unexpected chance for a stagnating model to achieve generalization, or if they are an unnecessary delay of generalization by the model being to rigid.

4.4 Training Loss Spikes

We saw in section 4.3 that grokking occurs because the model remains in a state of very low loss, almost without any change, and then the training loss spikes by several magnitudes. I was not able to find a clear explanation of these spikes, but their existence is the topic of a recent paper by Thilak et al. [7], where the authors argue the cause of these spikes to be due to an inductive bias of adaptive gradient optimizers.

As seen in figure 4.5, the training curves vary a fair amount for different train-validation-splits, especially considering that these experiments use different parts of the same dataset and the same random seed to initialize model weights. In Appendix B, similar plots for all other datasets are included.

It is clear that the operation used for generating the dataset has a big influence on the loss curves as it determines the loss landscape; still, it is remarkable that the spikes (as well as the flat valleys in between) occur for almost every dataset for almost every training percentage, even if they happen at different times and get less distinct later in the training process. At a train-validation-split of 60%, several different experiments' training loss curves oscillate instead of the usual consistent drop and delayed spike. I have no explanation for this behaviour and it might be an additional question for future research.

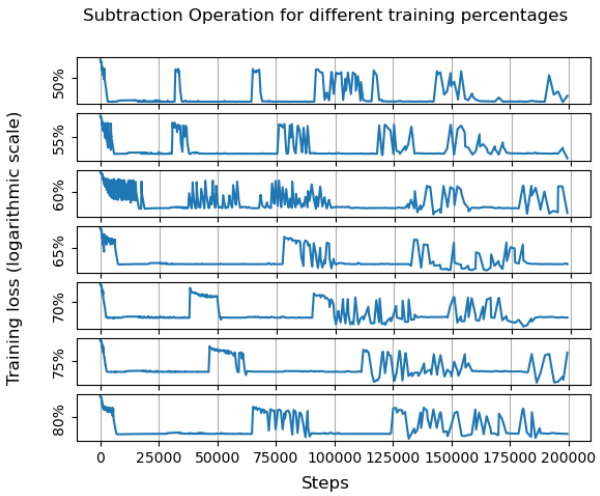


Figure 4.5: Broad comparison of different training loss curves (different parts of the same dataset, same random seed)

These training loss spikes, as well as the lingering in the local minimum in between, seem to happen very consistently. They happen even where generalization is not possible for any percentage of training data as seen in Figure 4.6. They also happen when grokking has occurred already and the validation accuracy is perfect, as can be seen in Figure 4.4. The optimizer will disturb the model after

some time, even if it is already located in the best local minimum. But still, the quality of the local minima of the model do matter, as not once in my experiments the model jumped to a minimum that generalized worse. In other words: There is no un-grokking.

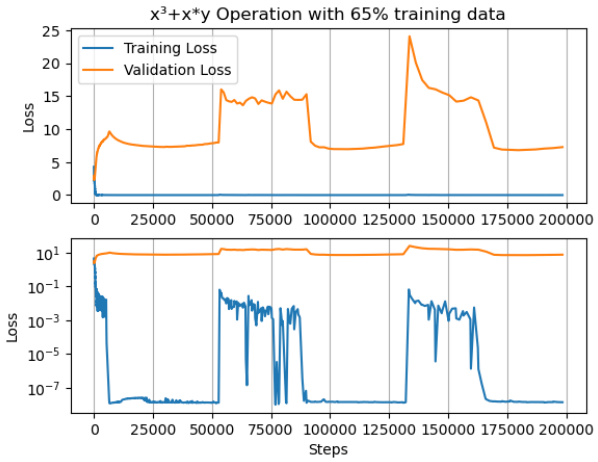


Figure 4.6

4.5 Irregular Validation Accuracy Curves

My experiments provided lots of data worth analyzing. In this section, I want to show some selected figures that depict edge cases and more messy situations, in an attempt to show grokking as something that is hard to define clearly and is a consequence of various factors.

In Figure 4.7, grokking is not happening all at once, but split up in two distinct steps. Otherwise, the learning behaviour looks like exemplary grokking as the validation loss as well as the validation accuracy are stable for the majority of the time. The first jump is not as steep as the second one, so it looks like grokking needs some wind up.

In Figure 4.8, there is quite good generalization from the start, very similar in appearance to the instantaneous generalization also called comprehension. For grokking to occur, there are still many optimization steps needed. The small training loss spikes before the actual grokking seem to have no influence at all on the validation loss and validation accuracy.

This is most likely due to the symmetry of the multiplication operator used to generate that dataset, as similar behaviour can be seen with the addition operation and, to a lesser extent, the squares-addition operation ($x^2 + y^2$). As Powers et al. mention in their work [1], the transformer model can just ignore position

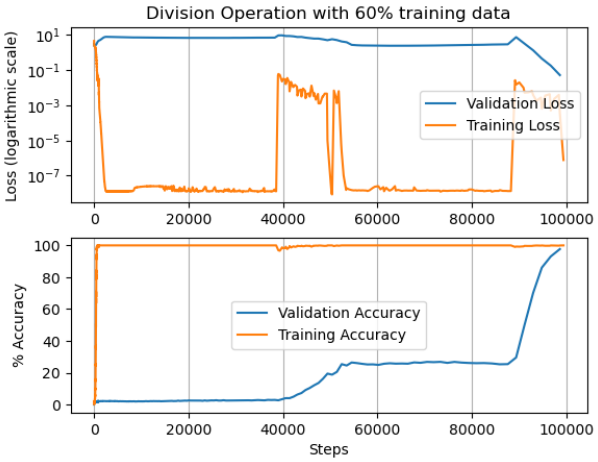


Figure 4.7

embeddings to explain all the duplicates in the dataset. This aspect of model architecture might account for the instantaneous high level of generalization on its own.

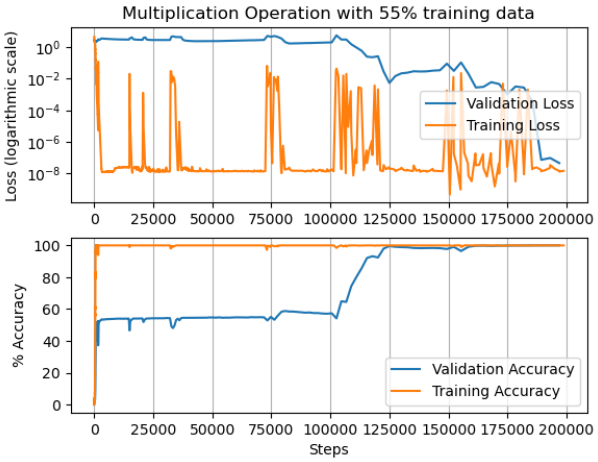


Figure 4.8

In Figure 4.9, after a high base level of generalization which might indicate a symmetrical operation, the validation accuracy rises approximately linearly and is almost perfect when the first grokking spike would have occurred at about 40000 steps. While generalization is still reasonably slow, this signifies that the model is not located in any local minimum. The training loss spikes definitely accelerate the changing of validation loss, but are not essential to get to better accuracy.

In Figure 4.10, no perfect generalization is achieved after $2 * 10^6$ steps. Still,

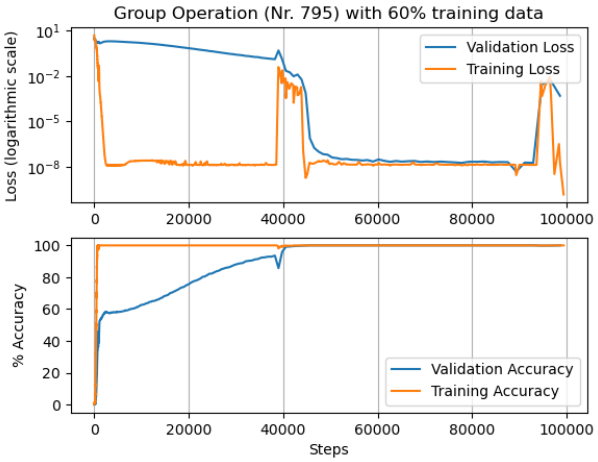


Figure 4.9

the validation accuracy is rising, though it is hard to say if the training loss spikes do or do not help the process. Especially after 50000 steps, the spaces between the training loss spikes seem flat, which would suggest that these plateaus are in fact different local minima and there is an almost unlimited amount of them. If on the other hand the training loss spikes do not help in generalization, this would be inconsistent with all other findings according to which these spikes may not be necessary for achieving good generalization capabilities, but certainly accelerate that process.

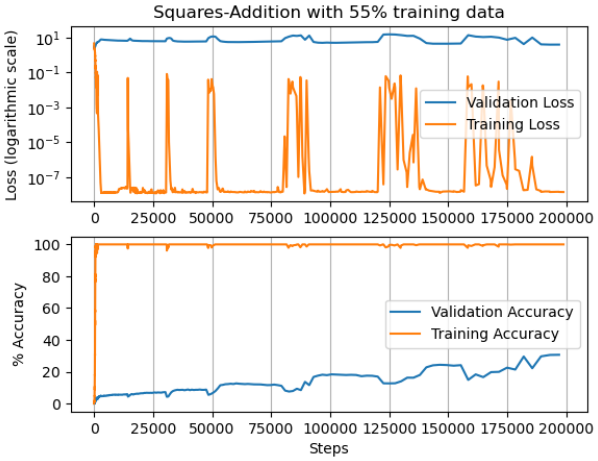


Figure 4.10

Chapter 5

Conclusion

In my experiments, replicating the original work of Powers et al. [1], and subsequent analysis thereof I have shown examples of grokking in different cases (including a newly incorporated dataset) and confirmed its connection to the amount of training data. Empirically, I have shown grokking to be bound to training loss spikes which coincides to the discovery of the "slingshot mechanism" by Thilak et al. [7]. The timing of these spikes varies a lot, is not consistent over different datasets and also gets less distinct over time; this only was observable by doubling the original experiments' initial time frame. It might be sensible to take even longer time frames into account, as the grokking-inducing instability of the models rather increased than decreased.

The phenomenon of grokking remains puzzling and its impact on machine learning theory hard to assess. In a case-by-case inspection of my experiments, several special cases of grokking could be observed, including: grokking split up in multiple steps; remaining at a validation accuracy well above chance level before jumping grokking-like to perfect accuracy; the validation accuracy steadily, but slowly rising until capped off by a grokking-like jump; the validation accuracy rising slowly with no sign of grokking. These are some phenomena any comprehensive theory of grokking has to explain.

Bibliography

[1] A. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra, "Grokking: Generalization beyond overfitting on small algorithmic datasets," 2022. [Online]. Available: https://arxiv.org/abs/2201.02177

[2] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep double descent: Where bigger models and more data hurt," 2019.

[3] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15 849–15 854, jul 2019. [Online]. Available: https://doi.org/10.1073%2Fpnas.1903070116

[4] Z. Liu, O. Kitouni, N. Nolte, E. J. Michaud, M. Tegmark, and M. Williams, "Towards understanding grokking: An effective theory of representation learning," 2022. [Online]. Available: https://arxiv.org/abs/2205.10343

[5] A. Gromov, "Grokking modular arithmetic," 2023.

[6] Z. Liu, E. J. Michaud, and M. Tegmark, "Omnigrok: Grokking beyond algorithmic data," 2022.

[7] V. Thilak, E. Littwin, S. Zhai, O. Saremi, R. Paiss, and J. Susskind, "The slingshot mechanism: An empirical study of adaptive optimizers and the grokking phenomenon," 2022.

[8] X. Davies, L. Langosco, and D. Krueger, "Unifying grokking and double descent," 2023.

[9] S. Peter, "Benchmarking the algorithmic reasoning of neural models," 2023. [Online]. Available: https://pub.tik.ee.ethz.ch/students/2022-HS/BA-2022-42.pdf

Appendix A

Reproducing the Experiment: Problems and Solutions

Reproducing the experiments of the original grokking paper [1] has turned out more complicated than initially anticipated. This is a recap of my process that might help anybody interested in running this code by themselves. A link to the original code can be found on in the footnotes[1].

The first challenge was setting the conda environment up correctly. The authors of the code did not specify which version of the python packages they used, and neither did they respond to comments on their Github page. I had to try out different versions of the python packages used based on when I assumed the original authors to have started working on their project. On the Github page for this thesis[2] I uploaded a working environment setup as a requirements.txt.

Thankfully, I was able to do computations on the Arton cluster from the ITET-department, and with it I got to use SLURM for the first time. As the experiments had to store quite an amount of data, running the experiments was not possible from my standard ETH-student account, so I had to use the shared online storage net-scratch, which in turn also required to re-install conda to also be located in the same online storage system. With the correct setting of paths, bash-jobs calling python scripts could be submitted on SLURM.

Running the experiments is done by calling the file "train.py" in the folder "scripts". Parameters could be passed on by additional arguments, while the important ones had to be figured out as the code was lacking in documentation. The parameter "datadir" had to be defined but was not used, while "logdir" was the place where the experiment data was stored in the end. All dataset, model and optimizer settings can be specified here. To achieve any form of generalization, the hyperparameters have to be tuned. I tried around for some time, mostly going by the parameters Powers et al. suggest in their original grokking paper [1]. The most crucial parameter is "weight_decay" for the AdamW optimizer, which

[1] Original Work by Powers et al. at https://github.com/openai/grok

[2] A Github Repository for files related to this thesis can be found at https://gitlab.ethz.ch/disco-students/hs22/investigating-grokking

I set to 0.01, while frequently varying "math_operator", "train_data_pct", "max_steps" and "random_seed". All the default parameters of the custom AdamW optimizer can be found in the project code.

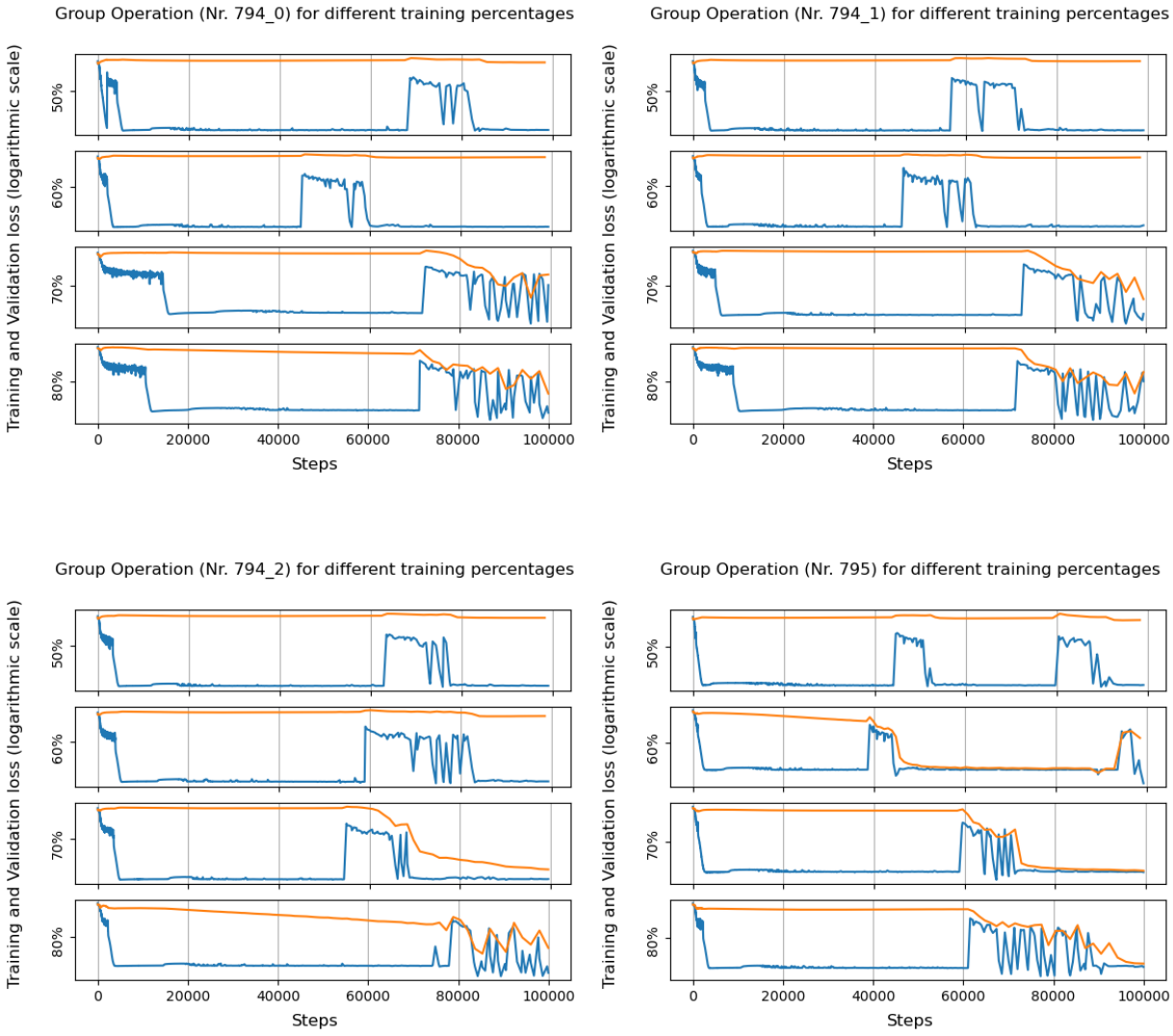
To be able to include datasets generated from the "Groups" database Simon Peter has created in his Thesis [9], I first converted single groups to CSV files, formatted them in a consistent way, and then changed the files "data.py" and "training.py" from the folder "grok" to be able to account for these new groups. These two files can be replaced to get the additional option to choose "custom" as "math_operator" while putting in the dataset name as a newly created argument. The code is flexible enough to incorporate groups of arbitrary size, as long as they were created with my own script. The two customized files as well as my script to extract CSV files from the "groups" database can be found on the Github repository[2].

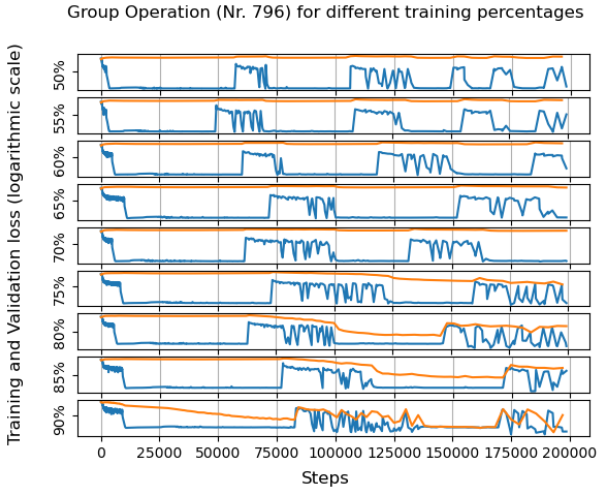
To generate plots using the code the authors provide, one can use the file "visualize_metrics.py", also located in the folder "scripts". As the input parameter, the folder one level above the folder with the experiment data has to be chosen or the script does not work. But only a single folder may be in that input folder, effectively requiring the experiment data to be saved in anotherwise empty folder. As some automatically generated paths did not work, I had to rewrite the code to fit to my folder structure, and I also added an additional argument for further use. There were other scripts in the "script" folder that also involved plotting data, but I could get none of them to work except "visualize_metrics.py". Unsurprisingly, I discourage using their python script to generate plots. Still, I added all plots generated with this method to the Github repository[2].

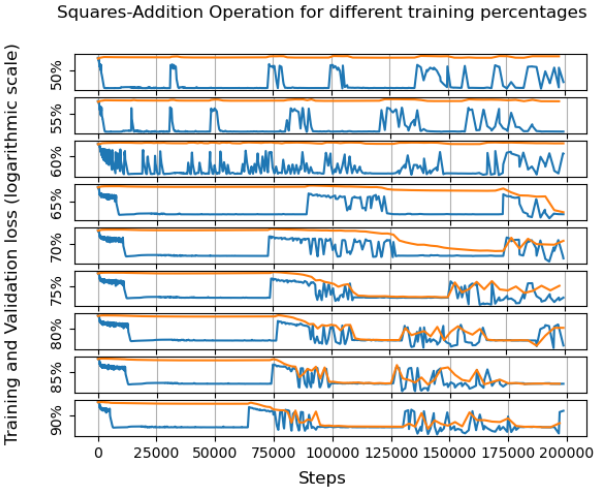
As I discovered later in the process, an easier way for me to analyze the data was to find the raw data and process it using my own python code. The CSV-file containing the raw data can be found at "/default/version_0/metrics.csv" inside the folder the experiment data was stored in. But there is an additional particularity: There might be several different subfolders inside the folder "default". In that case, the highest version has to be picked, or else the data is incomplete. I included my own code for extracting the data and generating plots, used for all figures in this thesis, in the Github repository[2] as a possible template.

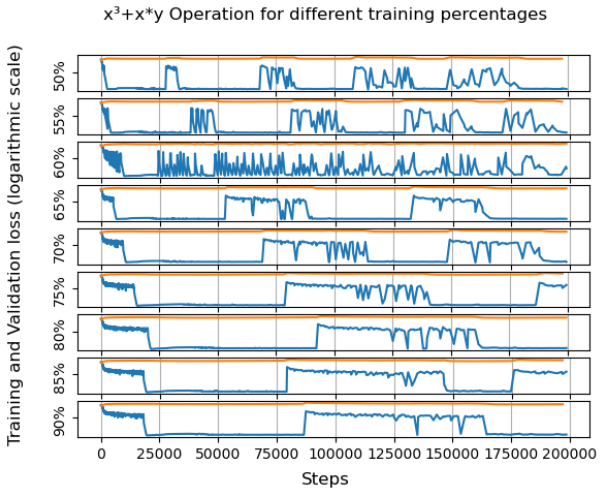
Appendix B

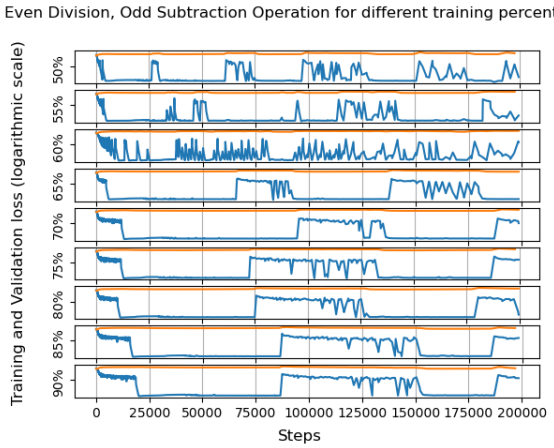
Additional Figures

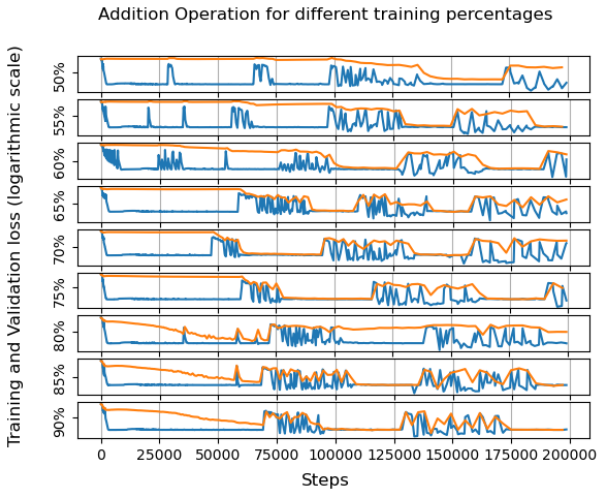


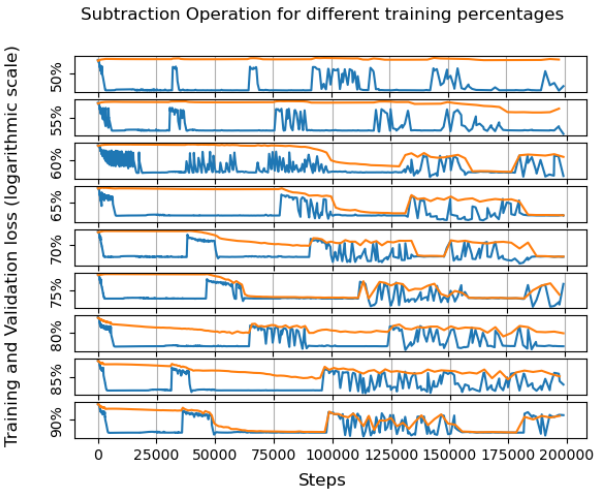

Group Operation (Nr. 796) for different training percentages
Training and Validation loss (logarithmic scale)
50%
55%
60%
65%
70%
75%
80%
85%
90%
0 25000 50000 75000 100000 125000 150000 175000 200000
Steps

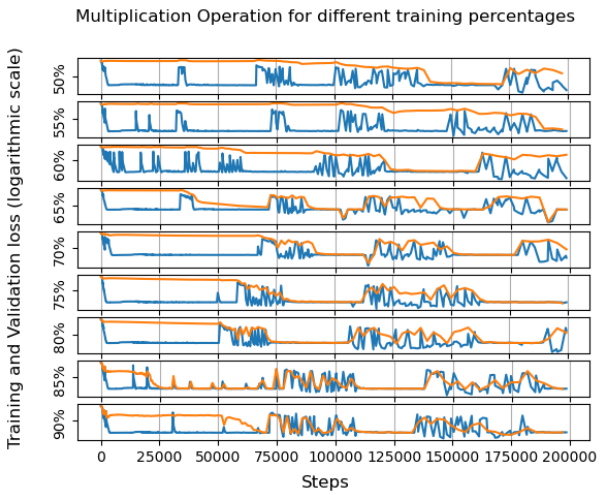

Squares-Addition Operation for different training percentages
Training and Validation loss (logarithmic scale)
50%
55%
60%
65%
70%
75%
80%
85%
90%
0 25000 50000 75000 100000 125000 150000 175000 200000
Steps

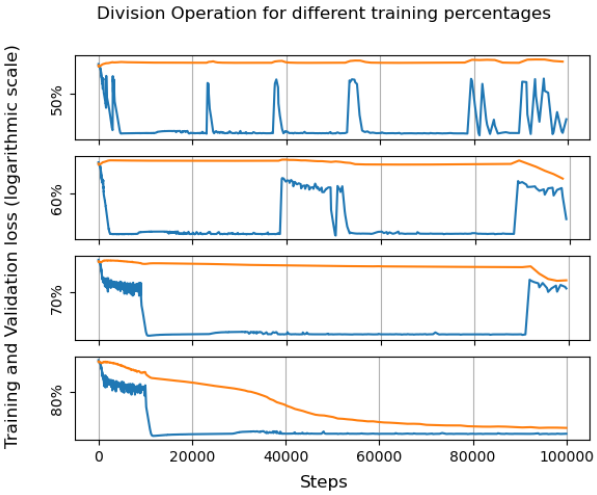

x³+x*y Operation for different training percentages
Training and Validation loss (logarithmic scale)
50%
55%
60%
65%
70%
75%
80%
85%
90%
0 25000 50000 75000 100000 125000 150000 175000 200000
Steps


Even Division, Odd Subtraction Operation for different training percentages
Training and Validation loss (logarithmic scale)
50%
55%
60%
65%
70%
75%
80%
85%
90%
0 25000 50000 75000 100000 125000 150000 175000 200000
Steps


Addition Operation for different training percentages
Training and Validation loss (logarithmic scale)
50%
55%
60%
65%
70%
75%
80%
85%
90%
0 25000 50000 75000 100000 125000 150000 175000 200000
Steps


Subtraction Operation for different training percentages
Training and Validation loss (logarithmic scale)
50%
55%
60%
65%
70%
75%
80%
85%
90%
0 25000 50000 75000 100000 125000 150000 175000 200000
Steps


Multiplication Operation for different training percentages
Training and Validation loss (logarithmic scale)
50%
55%
60%
65%
70%
75%
80%
85%
90%
0 25000 50000 75000 100000 125000 150000 175000 200000
Steps


Division Operation for different training percentages
Training and Validation loss (logarithmic scale)
50%
60%
70%
80%
0 20000 40000 60000 80000 100000
Steps