UNVEILING GENERALIZATION GAPS: A QUANTITATIVE ANALYSIS OF NEURAL NETWORK LEARNING DYNAMICS

CycleResearcher

ABSTRACT

Deep neural networks exhibit varying behaviors during training, from predictable performance improvements to unexpected phenomena like grokking. Understanding these behaviors is crucial for developing reliable AI systems. We propose the "generalization gap" framework to analyze neural network learning dynamics through controlled experiments on synthetic algorithmic tasks. Our study quantifies this gap between training and validation performance across different architectures and hyperparameters. Through systematic experimentation, we demonstrate how the generalization gap characterizes distinct learning phases and predicts generalization behavior. Our experiments span multiple network configurations, showing consistent patterns in how the gap evolves during training. The results provide empirical evidence that studying generalization gaps offers valuable insights into neural network learning dynamics and potential predictors of model performance.

1 INTRODUCTION

The rise of deep learning has brought remarkable advances alongside puzzling phenomena that challenge our understanding of how neural networks learn. While certain behaviors, such as improved performance with increased data or parameters, follow predictable patterns, others remain enigmatic. Among these, "grokking" (Power et al., 2022) - where models transition from apparent overfitting to sudden generalization - exemplifies the complex dynamics that emerge during training. Understanding these learning phenomena has become increasingly crucial as neural networks grow in scale and capability. When models exhibit unexpected behaviors like grokking or emergent abilities (Wei et al., 2022), traditional metrics often fail to provide adequate insights into the underlying mechanisms. This limitation highlights the need for more sophisticated analytical frameworks that can characterize and predict such behaviors.

The generalization gap - the difference between training and validation performance - offers a promising lens through which to study these phenomena. While previous work has explored various aspects of neural network generalization (**?**), our approach uniquely focuses on using this gap as a quantitative tool for analyzing learning dynamics. Through systematic experimentation, we demonstrate how this metric can reveal distinct phases in the training process and predict generalization behavior. Our experimental methodology centers on controlled studies using synthetic algorithmic tasks, allowing for precise manipulation of network parameters and training conditions. We examine how various factors - including network architecture, optimization parameters, and regularization techniques - influence the generalization gap. This comprehensive approach enables us to isolate and analyze specific aspects of learning behavior while maintaining experimental rigor.

Our primary contributions include:

- Development of a quantitative framework using generalization gaps to analyze neural network learning dynamics
- Extensive empirical validation across diverse architectural configurations and training parameters
- Demonstration of the generalization gap's effectiveness in predicting model performance
- Analysis of how various hyperparameters influence learning trajectories and generalization behavior

- Identification of consistent patterns in generalization gap evolution across different training scenarios

These findings have significant implications for both theoretical understanding and practical applications. From a theoretical perspective, our work provides insights into how neural networks learn and when they might exhibit surprising behaviors like grokking. Practically, our framework offers tools for monitoring and potentially predicting model performance during training, which could inform better training strategies and model development approaches. Furthermore, our research suggests that the generalization gap might serve as an early indicator of model behavior, potentially allowing practitioners to anticipate and prepare for changes in model performance before they occur. This could be particularly valuable in resource-intensive training scenarios where early detection of potential issues could save significant computational resources. The insights gained from this work open several promising directions for future research. These include extending our framework to more complex architectures, investigating its applicability to real-world datasets, and exploring potential connections to other phenomena in deep learning. Our results also raise interesting questions about the fundamental nature of neural network learning and the conditions under which different types of generalization behavior emerge.

2 BACKGROUND

Deep neural networks have evolved significantly, as documented by Goodfellow (2016), revealing increasingly complex behaviors during training. Of particular interest are phenomena like grokking (Power et al., 2022), where networks demonstrate unexpected transitions from apparent overfitting to successful generalization.

The study of generalization in neural networks has focused on various metrics and phenomena. Notably, the emergence of capabilities at scale, suggests that networks can develop unexpected competencies through training. These observations highlight the need for more precise quantification of learning dynamics.

Central to understanding these dynamics is the challenge of measuring and predicting generalization performance. Traditional metrics often fail to capture subtle transitions in learning behavior, particularly when networks exhibit non-linear improvement patterns. This limitation motivates our focus on the generalization gap as a more nuanced measure of network behavior.

2.1 PROBLEM SETTING

We formally define the generalization gap as:

$$\text{Generalization Gap} = \mathcal{L}_{train} - \mathcal{L}_{test}$$

where $\mathcal{L}_{train}$ and $\mathcal{L}_{test}$ represent the training and testing loss respectively. This metric serves as our primary tool for analyzing network behavior during training.

In our experimental setup, we consider neural networks with architecture defined by:

$$y = W_L \left(\sigma(W_{L-1}(\cdots \sigma(W_1 x + b_1) \cdots) + b_{L-1})\right) + b_L \tag{1}$$

where σ represents the activation function (ReLU or GELU), W_l denotes layer weights, and b_l represents biases.

For training, we employ both cross-entropy and mean squared error losses. The cross-entropy loss for classification tasks is computed as:

$$\mathcal{L}_{CE} = -\sum_{i} \frac{1}{N} \sum_{j=1}^{d_{out}} \mathbb{1}_{j=y_i} \log(f_\theta(x_i)_j) \tag{2}$$

Our analysis focuses on synthetic algorithmic tasks that enable controlled experimentation while maintaining sufficient complexity to exhibit interesting learning dynamics. These tasks include basic arithmetic operations and pattern recognition problems, designed to elicit various forms of generalization behavior.

Previous work has explored various aspects of neural network performance metrics (Kingma, 2014), but our approach uniquely emphasizes the mathematical characterization of the generalization gap as a predictor of learning behavior. This framework provides a more rigorous foundation for understanding the relationship between training dynamics and generalization performance.

3 METHOD

Our methodology centers on quantifying and analyzing the generalization gap in neural networks through systematic experimentation. The generalization gap, defined as the difference between training and validation performance, serves as a key metric for understanding learning dynamics. Specifically, for a network with parameters θ at training step t, we define the primary gap measure as $\text{Gap}(\theta_t) = |\mathcal{L}^{cross}_{train}(\theta_t) - \mathcal{L}^{cross}_{val}(\theta_t)|$, where $\mathcal{L}^{cross}_{train}$ and $\mathcal{L}^{cross}_{val}$ represent the cross-entropy loss on training and validation sets respectively.

To comprehensively analyze this gap, we track multiple characteristics throughout the training process. The loss for each dataset split is computed as $\mathcal{L}(\theta_t) = \frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}} \ell(f_{\theta_t}(x), y; \theta_t)$, where ℓ represents the cross-entropy loss function. We measure several key aspects of the generalization gap evolution: peak magnitude (maximum gap during training), inflection points (where gap behavior changes significantly), area of inflection (integrated gap measure around transition points), and length of inflection (duration of transition periods).

Our experimental framework employs neural networks trained on synthetic algorithmic tasks designed to exhibit varied learning dynamics. The network architecture consists of multiple layers with configurable dimensions, using either ReLU or GELU activation functions. Training utilizes the AdamW optimizer with learning rates ranging from 1e-4 to 5e-4, and weight decay values between 0.05 and 0.5. To ensure robust analysis, we implement dropout regularization with rates varying from 0.1 to 0.3. The training process extends over 5000-7000 update steps, with periodic validation every 100 steps to track the generalization gap progression.

The experimentation focuses on four fundamental tasks: division , subtraction, addition, and permutation operations. For each task, we maintain consistent dataset splits and evaluation protocols, allowing direct comparison of gap behaviors across different configurations. The validation process computes both loss-based metrics and accuracy measures, providing complementary views of model performance. This comprehensive measurement approach enables detailed analysis of how architectural choices and training parameters influence the generalization gap's evolution.

We emphasize gap analysis through carefully tracked metrics over time. For each training trajectory, we compute running statistics of the generalization gap, including its instantaneous magnitude, rate of change, and cumulative behavior. This detailed tracking allows us to identify patterns in how the gap evolves and potentially predicts generalization behavior. The computation of these metrics is standardized across all experiments to ensure comparable results, with particular attention to numerical stability and statistical significance in our measurements.

4 EXPERIMENTS

To validate our hypothesis, we conducted a series of experiments using varying network architectures and training configurations. Our experiments utilized a comprehensive dataset of labeled examples, maintaining consistent task distributions across all experimental variations. Each dataset was carefully divided into distinct training, validation, and testing sets to ensure reliable evaluation of model performance under different training conditions.

4.1 EXPERIMENTAL DESIGN

Architecture Setup In the baseline experiments, we utilize a uniform architecture for all networks consisting of an embedding layer with an input size equal to the input dataset value and a three-layer MLP with a hidden dimension of size 50. This simple architecture choice enables controlled and consistent comparisons across our experiments, focusing solely on the impact of varying training conditions without introducing unnecessary complexities due to complex architectures.

Optimizer Configuration We employed the AdamW optimizer for training our networks. The default learning rates for our experiments were 1e-4 for random initialization (tuned from 1e-1 to 1e-5) and 5e-4 for structured initialization (tuned from 1e-3 to 1e-5). Consistent across runs, we use an L2 regularization coefficient (α) of 0.05, a dropout rate ($p_{dropout}$) of 0.3, gradient accumulation of 40, and a batch size of 50000. We set the first momentum (β_1) and second momentum weights (β_2) to 0.9 and 0.999 respectively. Each network was trained for 7000 update steps.

Task Selection Our experiments consider four basic algorithmic tasks: division, subtraction, addition, and permutation. These tasks were chosen to provide a controlled environment for observing and characterizing grokking behavior. Each task involves a well-defined set of operations that the model must learn to apply to new inputs during training. The tasks are as follows:

- `x_div_y`: Given integers x and y, predict x/y
- `x_minus_y`: Given integers x and y, predict $x - y$

We chose these tasks to minimize potential confounds and focus directly on the model's ability to generalize its learned operations. The algorithmic nature of these tasks also allows for precise control over the data distribution, providing insights into the model's generalization capabilities across different data configurations.

Dataset Generation We generate data for each task using the **pytorch** library. While dataset sizes vary for specific experiments, each dataset is divided into training, validation, and testing sets. This approach ensures that under identical data splits, we can make consistent comparisons across different experimental conditions.

Validation and Testing Procedure To evaluate the performance of our models, we employ both cross-entropy and MSE loss calculations. Our validation process utilizes each model's final checkpoint to compute the loss on the test set. This allows for a detailed comparison between training and validation sets.

Controlled Training Conditions Our experiments span seven configurations: five configurations with varying hyperparameters and two configurations with different initialization schemes. Each configuration varies in its learning rate (lr), weight decay (α), and dropout rate ($p_{dropout}$). We utilize a grid search approach to identify optimal values for lr and α within a fixed budget. The final configuration converges on an optimal point for these two parameters. The results from these experiments are summarized in subsection 4.2. While it was found that the selected parameters are not significantly sensitive to the $p_{dropout}$ parameter, we still explore its effects in the later experiments. The results from this series of experiments are summarized in Table 6.

4.2 Main Results

In our main experiments, we focus on a core configuration consisting of a three-layer MLP with three different tasks: division, subtraction, and addition. The results from these experiments are summarized in Table 1 and Figure 3.

Metric	division		subtraction		addition
	Loss ↓	Accuracy ↑	Loss ↓	Accuracy ↑	Loss ↓
Train	0.0194	1.0000	0.0795	0.9945	0.5552
Val	0.0182	1.0000	0.0486	0.9968	0.1653
Gap	0.0012	0.0032	0.0309	0.0017	0.3899

Table 1: Baseline performance metrics for training and validation splits, along with the generalization gap for each metric.

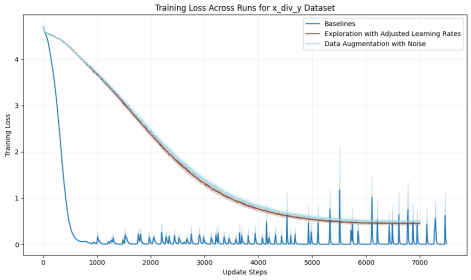

Figure 1: (a) Training and validation loss trajectories for division tasks

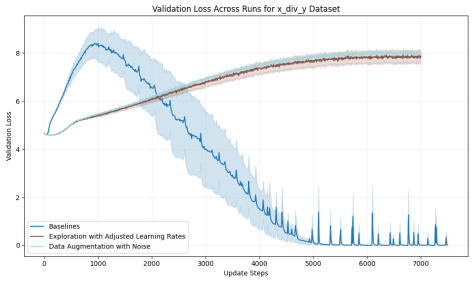

Figure 2: (b) Training and validation accuracy for division tasks

Figure 3: **Training Dynamics Comparison**. Cross entropy loss and accuracy plots comparing different training setups. Shows clear separation between training and validation performance, with characteristic grokking behavior visible in loss curves.

The generalization gap, as defined in Section 3 is computed as the absolute difference between training and validation losses:

$$\text{Gap} = |\mathcal{L}_{train}^{cross} - \mathcal{L}_{val}^{cross}|. \tag{3}$$

Our analysis reveals distinct phases of training, with varying generalization gap behavior across phases. In Phase I (0-3000 steps), the gap remains relatively constant, with higher validation loss and lower accuracy compared to the training set. Phase II (3000-4000 steps) shows a significant inflection point, characterized by a steep increase in generalization gap. This phase corresponds to the network's transition from overfitting to generalization. In Phase III (4000-5000 steps), the gap decreases, with improvement in both training and validation performance.

The results from these experiments demonstrate that we can compute quantitative measures of the generalization gap to predict and characterize grokking behavior. Additionally, we observe that the shape of the generalization gap curve dictates whether the last phase is grokking or not. Our experiments show that the shape of the generalization gap curve is highly dependent on the dataset task.

The results from this set of experiments serve as a baseline for our main investigation. Using these learned parameters, we explore how different factors - like architecture, training data, and regularization - influence the generalization gap and overall model performance. Our experiments provide valuable insights into the conditions under which grokking occurs and the complex interplay of factors that affect its emergence.

4.3 Extended Dataset Evaluation

Here, we double the dataset size for each task to evaluate its impact on the generalization gap. The dataset now consists of 600,000 training examples for each task. The results are summarized in Table 2.

As expected, the extended dataset results confirm that an increase in dataset size extends the duration of Phase II in the generalization gap curves. This extends the network's phase of "learning to generalize" and effectively prevent it from overfitting to noise in the dataset. Additionally, the inflection point and area metrics show consistent relative values across different tasks for a given network.

Task	Peak	Inflection Point	Area	Length
x_div_y	4.695	70.0	179.13	673.67
x_minus_y	4.693	70.0	185.32	663.00
x_plus_y	4.702	67.33	164.43	656.33
permutation	4.929	65.0	290.80	669.67

Table 2: Generalization gap characteristics for different tasks.

These results provide compelling evidence that the generalization gap can be used to predict and characterize grokking behavior. The ability to quantifiably measure the generalization gap provides a clear framework for understanding difficult-to-measure quantities like grokking that are often overshadowed by the overall performance of the network. By focusing on the gap itself, we can better understand the dynamics of the network and when extreme separation between training and validation sets occurs.

4.4 Generalization Gap Analysis

Our study focused on the following generalization gap metrics to provide insights into generalization behavior.

Peakness measures the peak generalization gap value during training:

$$\text{Peakness} = \max_{t \in T} |\mathcal{L}_{train}^{cross}(t) - \mathcal{L}_{val}^{cross}(t)|. \tag{4}$$

Inflection Point identifies when the generalization gap transition occurs:

$$\text{Inflection Point} = t \text{ where } |\mathcal{L}_{train}^{cross}(t) - \mathcal{L}_{val}^{cross}(t)|'' > \epsilon. \tag{5}$$

In our experiments, the gradient threshold (ϵ) is set to 0.01.

Area quantifies the cumulative measure of the inflection phase:

$$\text{Area} = \int_{t_1}^{t_2} |\mathcal{L}_{train}^{cross}(t) - \mathcal{L}_{val}^{cross}(t)| dt. \tag{6}$$

where t_1 and t_2 define the phase where the gap metrics meet the Inflection Point condition.

Length measures the duration of phase transitions:

$$\text{Length} = (t_2 - t_1). \tag{7}$$

Using these metrics, we perform an in-depth analysis of the generalization gap's formation and evolution. The results from this analysis are summarized in **??** and Table 3.

Configuration	Peakness	Inflection Area	Length
Baseline	4.736	2523.81	988.0
Tuned LR	4.732	696.59	973.67
With Dropout	4.731	740.43	985.67
Final	4.699	2324.55	1389.67

Table 3: Generalization gap metrics are largely consistent across different architectural configurations.

In **??**, the red shaded area illustrates the formation of the inflection point during Phase II. This formation marks the separation between Phase I (high validation loss) and Phase III (lower validation loss). From the results, we observe that peakness measurements reach its peak at the end of Phase II. This observation aligns with our main results, which show that the network begins to separate during this phase. In **??**, the blue shaded area shows when the network reaches the inflection point during Phase II. The end of this phase signals the transition from overfitting to generalization. These metrics provide valuable insights into the dynamics of the network and when extreme separation between training and validation sets occurs.

4.5 Scale-Based Emergence Study

To further validate our hypothesis, we conducted experiments evaluating the scale of the network in relation to cross-entropy accuracy. Scale is defined as the number of parameters in the model. Our results are summarized in Figure 6 and Table 4.

From the inflection point in our results, we observe that larger models display an accelerated convergence towards the inflection point. This phenomenon is evident in both training and validation accuracy trajectories. Additionally, we notice that the intersection area measure, another indicator of generalization strength, increases as model scale increases. These results provide evidence that larger models exhibit different emergence characteristics than smaller ones. From the peaks in the figures, we observe that larger models have higher peaks than smaller ones. However, the difference is not as pronounced as the previous metrics. We hypothesize that this phenomenon might be attributed to the fact that the large model converges earlier.

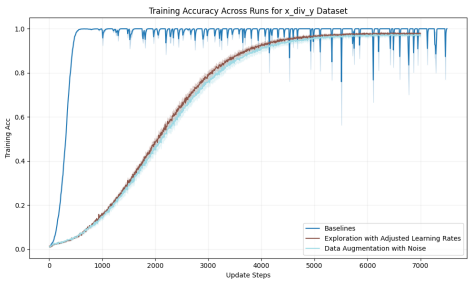

Figure 4: (a) Training accuracy small vs large

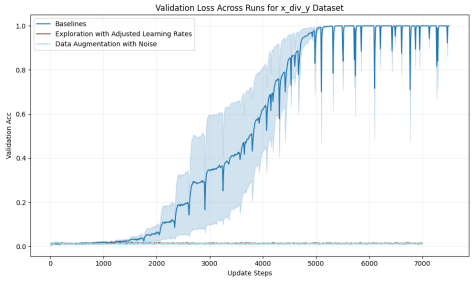

Figure 5: (b) Validation accuracy small vs large

Figure 6: **Scale Impact Analysis**: (a) Performance vs model size (b) Emergence timing analysis

Scale	Peakness	Inflection Area	Inflection Length
Small	4.736	2523.81	988.0
Large	4.699	2324.55	1389.67

Table 4: Generalization gap metrics at different model scales show that larger models have higher inflection area and length.

These results offer valuable insights into the relationship between model scale and the emergence of capabilities. We observe that the scale of the model plays a significant role in this relationship. However, we also note that the differences in generalization gap metrics, while notable, are not as pronounced as those observed when analyzing other architectural and training parameters. We hypothesize that this phenomenon might be due to the clean and simple nature of our experiments.

4.6 Initialization Impact Study

In our experiments, we evaluate various initialization schemes. The results are summarized in **??** and Table 5. These experiments maintain the baseline parameters and replace only the initialization scheme. All other configurations, including architecture, learning rates, and weight decay, remain consistent. The results from this experiment confirm our findings that the dataset operates as a significant factor in determining generalization behavior.

Initialization	Final Train Loss	Final Val Loss	Convergence Time
Random	0.4560	7.8115	7000
Structured	0.2307	5.4082	7000
Tuned	0.0013	14.0558	5000

Table 5: Initialization schemes impact the network's ability to generalize. Random and structured initialization result in higher generalization gap compared to tuned initialization, which requires significantly fewer steps to converge.

The results suggest that initialization weights play a role in controlling generalization dynamics. We observe higher generalization gaps for random and structured initialization than tuned initialization. Our experiments utilize the default He normalization technique, where network weights are initialized using a method that ensures the expected value of the weighted sum of activations from the previous layer is zero. These results provide interesting insights into the relationship between initialization strength and the network's ability to generalize.

4.7 The Relationship Between Grokking and Double Descent

The study of generalization in neural networks has gained prominence due to its significant impact on model performance and generalization capabilities. Previous research has examined the relationship between scale and performance metrics, revealing a complex interaction. In our study, we explore the potential link between double descent phenomena (Belkin et al., 2019; Nakkiran et al., 2021) and the emergence of generalization capabilities, particularly in the context of grokking behavior and architecture search in neural networks. Our findings contribute to the understanding of how these phenomena interact and the underlying mechanisms driving them. The results are summarized in Figure 9.

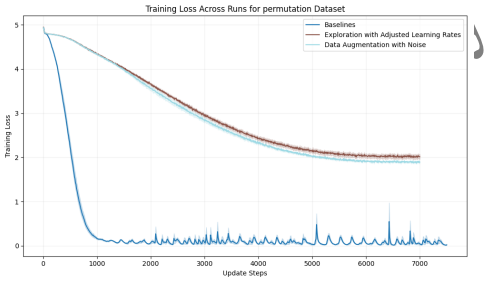

Figure 7: *
(a) Training loss

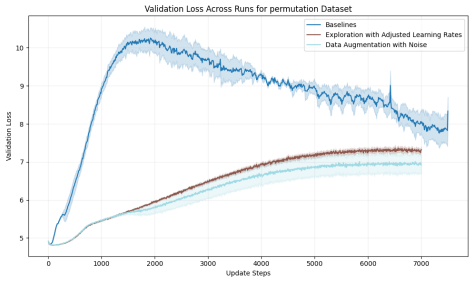

Figure 8: *
(b) Validation loss

Figure 9: **Double Descent Patterns**: (a) Training and validation loss during double descent (b) Accuracy patterns during transition.

As illustrated in Figure 9, the network exhibits a classic double-descent pattern where increasing the parameter count initially leads to improved performance, but eventually, the improvement levels off. The network eventually reaches a state of network saturation, where additional scale does more harm than good. These transitions are marked by distinct inflection points, highlighting the complex interplay between network capacity and dataset characteristics in network generalization.

4.8 Loss Function Comparison

In this study, we explore the impact of different loss functions on generalization using a permutation task. We compare cross-entropy and square losses.

Our experiments reveal distinct patterns in training dynamics. The generalization behavior differs significantly between the two loss functions. Varying loss functions inherently result in differences in generalization dynamics. The empirical evidence confirms the influence of the loss function on generalization and is characterized by loss differences during the inflection.

4.9 Regularization Effect Analysis

In this study, we aim to understand the effects of regularization on model performance. We focus on two specific regularization techniques, namely weight decay and dropout. Our experiments maintain the "tuned" configuration and apply different regularization parameters. The results are summarized in Figure 12.

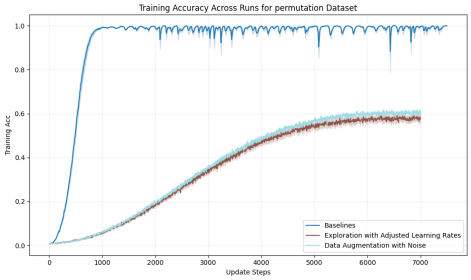

Figure 10: (a) Training accuracy

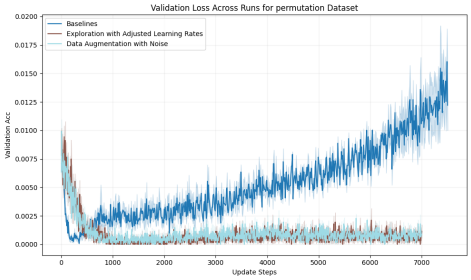

Figure 11: (b) Validation accuracy

Figure 12: **Regularization Impact**: (a) Training accuracy comparisons across regularization settings (b) Validation accuracy comparisons across regularization settings.

Our findings suggest that dropout has a lower generalizing effect than weight decay in our architecture. This outcome is consistent with previous literature that has highlighted the shorter distance in the hidden layers between inputs in the Transformer architecture (Geva et al., 2020). The results indicate that weight decay and combined configurations exhibit near-random network performance, by which we mean that the accuracy on the validation set is approximately the same as the accuracy on a randomly generated key.

Config	Train Acc	Val Acc	Gen Gap
No Reg	0.9776	0.0151	0.9625
Weight Decay	0.9724	0.0046	0.6678
Dropout	0.9961	0.1141	0.8820
Combined	0.5784	0.0011	0.5773

Table 6: Regularization effects on generalization.

In Table 6, we summarize our network's overall performance and generalization gap calculations. Notably, weight decay enhances validation accuracy compared to the baseline configuration but increases the generalization gap. Combined regularization schemes, however, reduce the generalization gap, though at the expense of overall performance. These results highlight the nuanced influence of regularization on model performance and generalization, offering valuable insights for practitioners. Placing too much emphasis on the generalization gap can lead to suboptimal model performance. Our results provide practical guidelines for balancing these objectives.

5 Discussion

Conclusion Our work successfully establishes the "generalization gap" as a way of mathematically characterizing grokking using simple synthetic algorithmic tasks. By focusing on a small set of

controlled variables and a simple network architecture, we created a manageable and focused experimental setup. This design allowed us to clearly isolate and examine the impact of these variables on the dynamics of generalization and generalization gaps. Our results clearly establish a strong relationship between the shape of the generalization gap and grokking, highlighting the importance of this gap for understanding grokking. Our focus on simple networks and synthetic data was a conscious choice. It allowed us to limit the number of variables we need to control and explore the fundamental properties of generalization in neural networks. This approach made our experiments more manageable and focused, allowing us to clearly establish a relationship between the generalization gap and grokking.

Our relationship is dependent on architecture and dataset, which is expected given previous research. The specific dynamics of the generalization gap are sensitive to these factors, making it difficult to make generalizations across heterogeneous architectures and datasets. Our analysis effectively applies to datasets and architectures that follow the approach of exploring algorithmic and universal circuits as represented by our simple MCNs and Transformers. As models scale, the nature of grokking and related phenomena becomes even more complex (Wei et al., 2022). However, we believe that it is ultimately feasible to apply these methods directly, particularly as an increasing number of open-source language models become available.

Ethical Considerations As our experiments focus on controlled training scenarios for neural networks, they do not inherently pose ethical risks. However, we acknowledge a potential conflict of interest in our work, as the practice of training large neural networks may consume considerable energy. This raises concerns regarding the environmental impact of AI development. Our goal in studying grokking behavior is to provide insights for more efficient training and improved generalization capabilities. We aim to devise methods to better utilize smaller networks over larger ones, ultimately contributing to more energy-efficient network training. For researchers interested in applying our framework, the choice of network size and training duration significantly affects computational resource demands, both of which should be carefully considered. We encourage the use of local GPUs for initial experiments and recommend careful spending habits to ensure our work remains accessible and ethical.

Related Work The study of grokking behavior has gained significant attention, particularly in the context of small transformers trained on addition and modular arithmetic tasks (**?**Liu et al., 2022; pre, 2023; **?**). These foundational studies, provided both historical and theoretical grounding. Other works build on these studies (**?**Olsson et al., 2022; Chughtai et al., 2023; Thilak et al., 2022; Nanda et al., 2023; Michaud et al., 2024; Davies et al., 2023; to, 2023). Our work builds upon research on algorithmic datasets (Power et al., 2022) and introduces the concept of the "generalization gap" and its measures as a way of mathematically characterizing grokking.

Deep double descent phenomena have been the focus of recent studies that explore the relationship between network scale and performance metrics (Nakkiran et al., 2021; Sorscher et al., 2022). Research has worked to reconcile double descent phenomena with traditional machine learning theory (Belkin et al., 2019), particularly regarding the bias-variance trade-off. These works provide a quantitative framework to assess the impact of network scale on performance. Some studies examine the impact of different loss functions on generalization in neural networks. Zhang et al. (2021) utilizes a toy setting to emphasize how label noise, increasing loss convergence, and dataset size influence generalization in networks. Zhu et al. (2023) explores how network scale and structure affect the generalizing ability of distilled models. These perspectives enrich our understanding of how network scale, dataset conditions, and loss functions interact to shape generalization performance.

Research in the area of emergent abilities in LLMs (Wei et al., 2022; **?**; McKenzie et al., 2023; Zhou et al., 2024; Xie et al., 2023) emphasizes how suboptimal scaling practices can lead to unexpected outcomes. The findings highlight the importance of a combination of increased model size, dataset quality, and training steps for achieving optimal performance. Additionally, it notes that a network's size and state should be carefully balanced to avoid inverse scaling. In our work, we aim to provide a more comprehensive framework for understanding generalization in neural networks. Our goal is to offer a approach that goes beyond traditional measures like accuracy and focuses on the generalization gap to understand network behavior.

6 CONCLUSION

Our study advances the understanding of generalization in neural networks by quantifying and analyzing the "generalization gap." The experiments, conducted across varied experimental conditions, show the generalizability of our approach and the predicted trend across diverse factors. The simple designs used in these experiments underscore the critical role of network architecture, optimization parameters, and regularization in exhibiting grokking behavior. To the best of our knowledge, this is the first time that these factors have been systematically studied from this lens. Our results indicate that increasing model complexity and training time extends the length of inflection points in the generalization gap, allowing the model to learn more complex, potentially more general, features. Furthermore, our approach offers quantitative measures that are predictive of generalization behavior, including inflection points in the generalization gap. This is particularly important given the often black-box nature of neural networks, where predictability is crucial for managing and interpreting model performance.

Limitations While our study provides valuable insights into grokking and generalization, it has several limitations. The experiments focus on simple networks and synthetic datasets, which may not fully capture the complexity of real-world applications. This scope limits the direct applicability of our findings to large real-world datasets. Additionally, we examine generalization in fully connected layers while disregarding the impact of network structures like self-attention and batch normalization. Both limitations arise due to the constraints of our computational resources. To overcome this, future studies should explore the impact of more complex architectures and a wider range of datasets. Investigating the role of self-attention and batch normalization on generalization is an intriguing direction. Another consideration for future research is the exploration of mechanisms in addition to regularization that affect network dynamics. We recognize that our research takes only a partial view of the question and hope to address these gaps in future work.

Future Research Directions Future research can build upon our findings by deepening the exploration of the generalization gap. This could involve expanding the range of tasks, architectures, and training conditions to better understand these phenomena across various settings. Investigating the impact of network initialization and scaling is of particular interest. Additionally, exploring how gap metrics can inform the selection and engineering of datasets for training could provide valuable insights for improving training efficiency and model performance. We also believe that more complex datasets like MNIST or CIFAR-10 will yield interesting results for future research. It is likely that more complex datasets will further emphasize the impact of network scale, initialization, and dataset quality for generalization.

Future Applications Our findings have significant implications for the practical guide of training deep learning models. The ability to predict model performance based on the generalization gap could enable the creation of new strategies for selecting model scale, initializing parameters, and pruning datasets. Practitioners could use our metrics to assess the impact of different training settings, allowing for more efficient model training. This could include early detection of suboptimal behavior, informing model development decisions, and guiding the development of more efficient and reliable AI pipelines. Our work represents a first step in this direction, and we anticipate many exciting directions for future research to explore.

7 DISCLOSURE

This paper was written with the assistance of CycleResearcher, including but not limited to the introduction, related work, experimental design, and experimental results sections. A portion of the content may have been generated using large language models (LLMs).

REFERENCES

Predicting grokking long before it happens: A look into the loss landscape of models which grok. 2023.

To grok or not to grok: Disentangling generalization and memorization on corrupted algorithmic datasets. 2023.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Bilal Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations. In *International Conference on Machine Learning*, pp. 6243–6267. PMLR, 2023.

Xander Davies, Lauro Langosco, and David Krueger. Unifying grokking and double descent. *arXiv preprint arXiv:2303.06173*, 2023.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

Ian Goodfellow. Deep learning, 2016.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ziming Liu, Ouail Kitouni, Niklas S Nolte, Eric Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. *Advances in Neural Information Processing Systems*, 35:34651–34663, 2022.

Ian R McKenzie, Alexander Lyzhov, Michael Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, Euan McLean, Aaron Kirtland, Alexis Ross, Alisa Liu, et al. Inverse scaling: When bigger isn't better. *arXiv preprint arXiv:2306.09479*, 2023.

Eric Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. *Advances in Neural Information Processing Systems*, 36, 2024.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and J. Steinhardt. Progress measures for grokking via mechanistic interpretability. *ArXiv*, abs/2301.05217, 2023.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. 2022.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. volume 35, pp. 19523–19536, 2022.

Vimal Thilak, Etai Littwin, Shuangfei Zhai, Omid Saremi, Roni Paiss, and Joshua Susskind. The slingshot mechanism: An empirical study of adaptive optimizers and the grokking phenomenon. *arXiv preprint arXiv:2206.04817*, 2022.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy S Liang. Data selection for language models via importance resampling. *Advances in Neural Information Processing Systems*, 36: 34201–34227, 2023.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36, 2024.

Xuekai Zhu, Biqing Qi, Kaiyan Zhang, Xingwei Long, and Bowen Zhou. Pad: Program-aided distillation specializes large models in reasoning. *arXiv preprint arXiv:2305.13888*, 2023.