Visualizing Grokking: The Impact of Dynamic Weight Decay on Training and Test Loss Convergence

Anik Biswas
Dept. of Math and Computer Science
Hobart and William Smith Colleges
Geneva, USA
ORCID: 0000-0002-1788-1321
anik.biswas@hws.edu

Md Khairul Islam
Dept. of Math and Computer Science
Hobart and William Smith Colleges
Geneva, USA
ORCID: 0009-0007-5868-0378
khairul.robotics@gmail.com

Hanqing Hu
Dept. of Math and Computer Science
Hobart and William Smith Colleges
Geneva, USA
hu@hws.edu

***Abstract*—Grokking,the phenomenon that neural networks generalize well after a long time of overfitting, has been actively studied in deep learning . In this work, we investigate the role of weight decay in speeding up grokking dynamics on modular arithmetic tasks. This paper presents and analyzes dynamic weight decay, a new approach that dynamically adjusts the weight regularization strength during training and is able to cut the time to reach generalization by a large margin. We investigate the interplay between sparsity, weight changes, and test loss convergence by a detailed study of input/output embeddings, attention weights, and loss trajectories. Our results show a high initial weight decay with dynamic adjustment strategy leads to faster grokking compared to static weight decay. These findings have broad implications in optimizing neural network training for tasks that require both memorization and generalization.**

***Index Terms*—Transformer model, weight decay, dynamic regularization, attention mechanism, positional embeddings, scaled dot-product attention, neural network training, AdamW optimizer, embedding visualizations, model regularization, heatmap analysis, deep learning.**

I. Introduction

Grokking in neural networks is a process where the model learns the pattern of a dataset after random probabilistic generalization, well only after prolonged overfitting, has gained immense attention in AI. From the initial observations on the modular arithmetic dataset, grokking has been subsequently used for algorithmic datasets, hierarchical reasoning, and semantic analysis tasks [1, 2]. Such generalization often results from an interplay between training dynamics, memorization, and structural simplicity [3, 4]. Weight decay, being one variant of L2-regularization, acts in a manner to keep the solutions simpler by adding penalty to the model's cost function. Previous works indicate that static weight decay helps neural networks move from solutions focused on memorization to generalized ones [5, 6]. However, such static regularization methods cannot support evolving training dynamics optimally; therefore, an adaptive method can be further explored. This paper introduces dynamic weight decay as a regularization technique where the weight decay coefficient is adaptively adjusted through training. In contrast to static weight decay, the dynamic approach is sensitive to the progress of the network itself and aids the process of convergence to generalization. We also investigated the interaction of sparsity, weight dynamics, and loss trajectories in the context of this method using modular arithmetic tasks like (a+b) mod(113). This contribution specifically shows that dynamic weight decay can be a factor to decrease the time taken for grokking to achieve, and shows empirical evidence on dynamic adjustments linked with improvements in test loss convergence. Our contributions include a demonstration of how dynamic weight decay significantly reduces the time to achieve grokking, an exploration of its impact on neural representations, and empirical evidence that links dynamic adjustment with improved test loss convergence.

II. Related Work

The original formulation of grokking emerged from studies on modular arithmetic tasks, where delayed generalization was linked to weight decay and structural regularization [1,2]. Recent works have extended this phenomenon to more complex datasets and architectures, highlighting its relevance across neural network training paradigms [3,4]. It has been realized that weight decay is quite central to grokking: it encourages circuit efficiency and may lead to the emergence of sparse subnetworks, according to some studies [5, 6]. On the other hand, the existing methods rely mainly on static weight decay schedules. This may not completely harness the evolving dynamics during training [7]. In grokking contexts, while sparsity and quality of representation are well explored, dynamic regularization is hardly explored. Building on this body of work, our study proposed and analyzed dynamic weight decay, which indicated how model efficiency and generalization speed can be potentially be improved by adaptive regularization.

III. Methodology

Here we describe the systematic approach to solving the modular arithmetic task $(a + b) \mod (113)$ [1, 2, 9]. The methodology integrates modular arithmetic task formulation,

dataset construction, model architecture, training strategies, weight decay regularization, and visualizations.

A. Dynamic Weight Decay

Unlike static weight decay, which maintains a constant λ throughout training, dynamic weight decay modifies λ based on stages of training to better align with the network's learning dynamics.

1) Adjustment Strategy: Our dynamic adjustment strategy involves the following steps:

1) Initial Phase: Apply a high weight decay coefficient to encourage sparsity and prevent early overfitting.
2) Intermediate Phase: Gradually reduce λ to allow the network to fine-tune weights without excessive regularization.
3) Final Phase: Stabilize λ to a lower value to maintain generalization without hindering performance.

The transition between phases can be governed by metrics such as validation loss plateauing or a predefined number of epochs.

Dynamic weight decay adjusts λ_t during training:

$$\lambda_t = \begin{cases} \lambda_{\text{initial}}, & t < t_{\text{warmup}} \\ \lambda_{\text{final}} + (\lambda_{\text{initial}} - \lambda_{\text{final}}) \cdot \left(1 - \frac{t - t_{\text{warmup}}}{T - t_{\text{warmup}}}\right), & t \geq t_{\text{warmup}} \end{cases} \tag{1}$$

B. Dataset Construction

We evaluate the effectiveness of dynamic weight decay on modular arithmetic tasks, specifically $(a + b) \mod 113$. The task is to compute:

$$\text{Output} = (a + b) \mod (113) \tag{2}$$

where $a, b \in [0, 112]$. This yields $113^2 = 12,769$ unique data samples. Each input sequence is represented as:

$$\mathbf{x}_i = [a, b, 113], \quad y_i = (a + b) \mod (113) \tag{3}$$

The inclusion of the special token 113 ensures a consistent input length of three tokens for all samples, enabling compatibility with the transformer model's fixed input requirements.

C. Dataset Construction

The dataset comprises all pairs (a, b) where $a, b \in [0, p-1]$, resulting in p^2 unique samples. It is split into training and testing subsets, with 70% allocated for training and the remaining for testing. This ensures a diverse representation for both phases of model evaluation.

D. Model Architecture

The transformer model is designed specifically for the modular arithmetic task [4, 6, 10]. It consists of the following components:

1) Input Embedding Layer: The input embedding layer maps input tokens into a higher-dimensional space for representation. Each input token is assigned a d_{model}-dimensional vector, facilitating downstream processing by the transformer model [7, 10].

2) Positional Embedding: To encode the order of input tokens, a positional embedding is added to the token embeddings. This enables the model to capture sequence information essential for modular arithmetic tasks.

3) Attention Mechanism: The transformer utilizes scaled dot-product attention to compute relationships between tokens. By assessing the compatibility of query and key vectors, attention weights are calculated and used to aggregate value vectors, allowing the model to focus on relevant parts of the input sequence [7, 11].

4) Feed-forward Layer: Each transformer block contains a feed-forward layer that applies a two-layer dense network with ReLU activation. This layer provides nonlinear transformations to enhance feature extraction and improve the model's representational power.

5) Output Projection: The output projection layer maps the final hidden state of the transformer to the output space. It applies a linear transformation followed by a softmax function, converting the model's learned representations into probabilities over the output vocabulary.

Layer (Type: Depth-Idx)	**Output Shape**	**Param #**
HookedTransformer	[1, 3, 113]	–
– Embed: 1-1	[1, 3, 128]	14,592
– HookPoint: 1-2	[1, 3, 128]	–
– PosEmbed: 1-3	[1, 3, 128]	384
– HookPoint: 1-4	[1, 3, 128]	–
– ModuleList: 1-5	–	–
—- TransformerBlock: 2-1	[1, 3, 128]	–
—— HookPoint: 3-1	[1, 3, 128]	–
—— Identity: 3-2	[1, 3, 128]	–
—— Identity: 3-3	[1, 3, 128]	–
—— Identity: 3-4	[1, 3, 128]	–
—— Attention: 3-5	[1, 3, 128]	66,048
—— HookPoint: 3-6	[1, 3, 128]	–
—— HookPoint: 3-7	[1, 3, 128]	–
—— Identity: 3-8	[1, 3, 128]	–
—— MLP: 3-9	[1, 3, 128]	131,712
—— HookPoint: 3-10	[1, 3, 128]	–
—— HookPoint: 3-11	[1, 3, 128]	–
– Unembed: 1-6	[1, 3, 113]	14,577
Total Params		**227,313**
Trainable Params		**227,313**
Non-Trainable Params		**0**
Input Size (MB)		0.00
Forward/Backward Pass Size (MB)		0.01
Params Size (MB)		0.12
Estimated Total Size (MB)		**0.13**

TABLE I
MODEL ARCHITECTURE BY LAYERS AND AND PARAMETERS

E. Training Procedure

The model is trained using optimizer, AdamW with the following hyper-parameters:

- Learning rate: 0.001
- Checkpoint : 1000 epochs
- Epochs: 25000
- Weight decay schedule: drops by 0.04 per epoch

1) Loss Function: The task is modeled as a classification problem using cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log P(y_i|\mathbf{x}_i) \tag{4}$$

2) Optimizer: The AdamW optimizer updates weights using:

$$w_{t+1} = w_t - \eta\left(\nabla_w \mathcal{L} + \lambda \cdot w_t\right) \tag{5}$$

where η is the learning rate and λ is the weight decay coefficient.

F. Analysis Metrics

To assess the impact of dynamic weight decay, we analyze:

- Training and Test Loss Convergence: Monitoring how quickly and effectively the model converges on training and test loss, on logarithmic and linear scale.
- Sparsity Levels: Measuring the sparsity of weight matrices throughout training.
- Weight Dynamics: Tracking changes in weights, including magnitude and direction.
- Neural Representations: Examining input/output embeddings and attention weights as heat maps.

IV. Results

This section provides a detailed analysis of the model's sparsity patterns, weight distributions, and training dynamics, which are critical in understanding how dynamic weight decay accelerates grokking while maintaining performance.

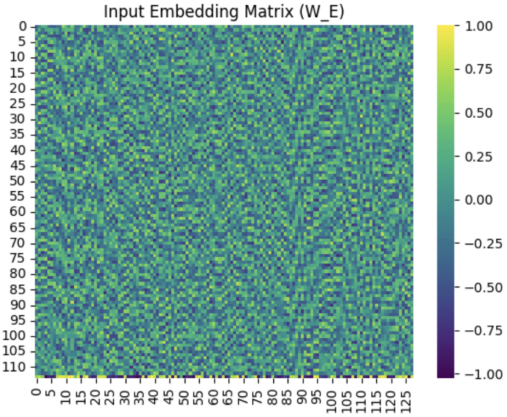

Fig. 1. Input Embedding Matrix (W_E). Shape: (114, 128). This Heat-map visualizes the normalized input embedding values.Sparsity:0.47%

A. Sparsity Analysis

1) Input Embedding Matrix (W_E): :

- *Observation*: The input embedding matrix (W_E) exhibits 47% sparsity (Fig. 1). This sparsity enables the model to efficiently encode input features while reducing redundancy [1, 5, 16].
- *Implication*: Such sparsity aligns with previous studies [1], [2], which indicate that reduced connections can enhance task-specific interpretability without compromising performance.

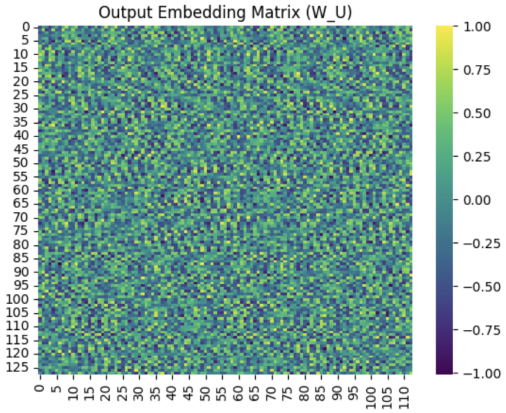

Fig. 2. Output Embedding Matrix (W_U). Shape: (128, 113). This Heat-map visualizes the normalized output embedding values. Sparsity: 0.57%.

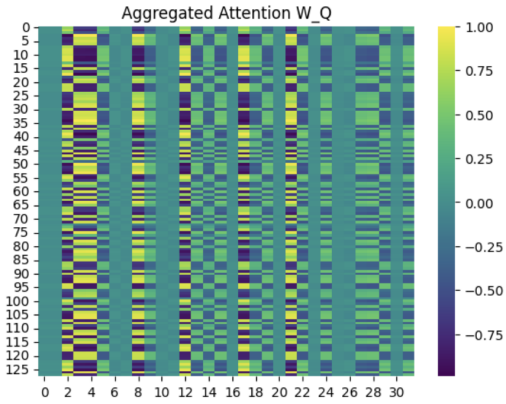

Fig. 3. Aggregated Attention Weight Matrix (W_Q) . Shape: (4, 128, 32). Sparsity of (W_Q): 0.00%

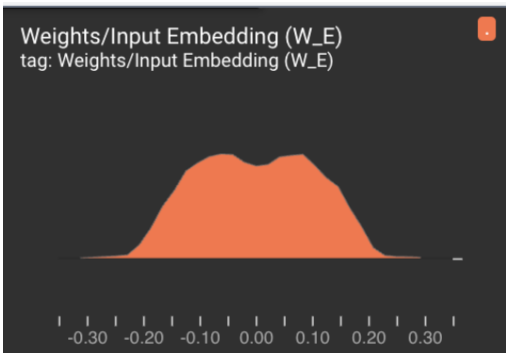

Fig. 4. Input Embedding Weight Distribution

2) Output Embedding Matrix (W_U): :

- *Observation*: The output embedding matrix (W_U) demonstrates 57% sparsity (Fig. 2), representing a more compact representation than W_E [6, 7, 11].
- *Implication*: This sparsity likely reduces overfitting, supporting findings in [3], which argue for the role of compact representations in achieving generalization.

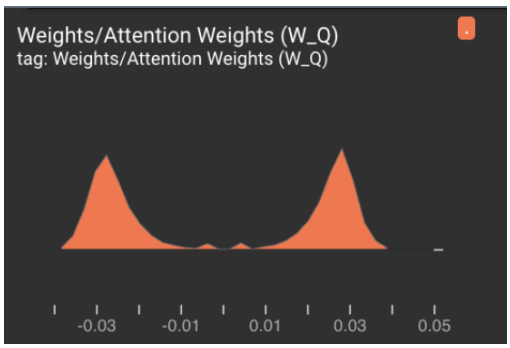

Fig. 5. Attention Weight Layer Weight Distribution

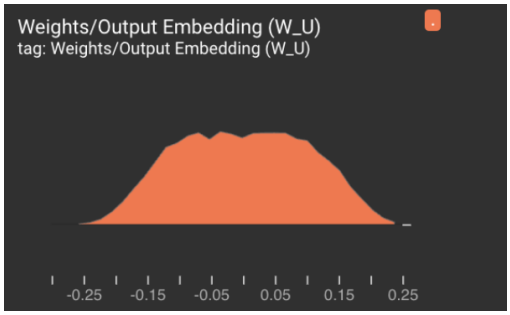

Fig. 6. Output Embedding Weight Distribution

Fig. 7. Legend of Fig. 8,9

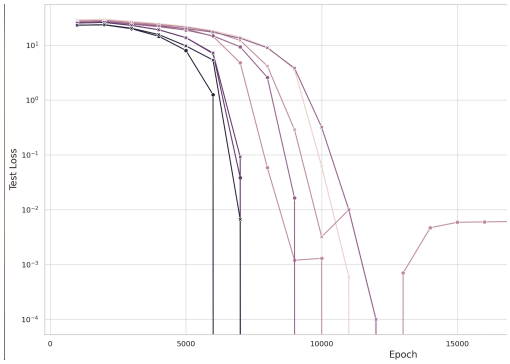

Fig. 8. Logarithmic loss

3) Aggregated Attention Weights (W_Q): :

- *Observation*: The aggregated attention weights (W_Q) exhibit a uniformly dense structure with sparsity 0% (Fig. 3) [4, 5].
- *Implication*: Dense structures facilitate robust feature extraction and global dependency modeling, consistent with the transformer-based research outlined in [4].

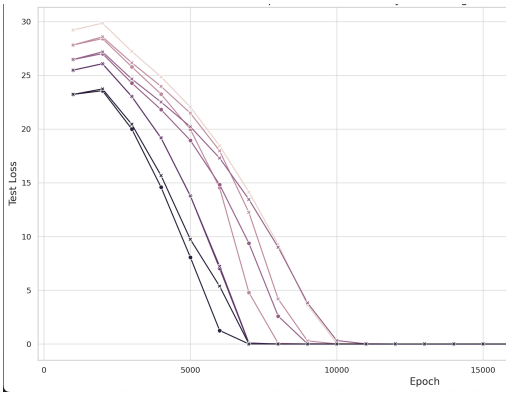

Fig. 9. Linear loss

B. Weight Distributions

1) Input Embedding Distribution (W_E): :

- *Observation*: Bimodal distributions in W_E (Fig. 4) reveal adaptive weight sparsity .
- *Insight*: These distributions support adaptive feature isolation mechanisms, aligning with findings in [5] on interpretability-driven sparsity.

2) Aggregated Attention Weight Distribution (W_Q): :

- *Observation*: Compact and narrow distributions in W_Q (Fig. 5) reflect the importance of uniform dense attention in enabling robust interactions.
- *Insight*: This supports prior transformer literature [6] [11] .

3) Output Embedding Distribution (W_U): :

- *Observation*: A broader spread in W_U (Fig. 6) indicates diverse roles in capturing latent representations.
- *Insight*: This finding aligns with [7], emphasizing adaptive sparsity for better generalization [6, 17].

C. Training Dynamics

1) Loss Convergence: :

- *Observation*: In the logarithmic scale plot (Fig. 7), dynamic weight decay exhibits a sharp drop in test loss compared to static decay, highlighting its efficacy in achieving faster convergence during early training phases [8, 14]. In the linear scale plot (Fig. 8), the smooth stabilization of test loss further illustrates the effectiveness of dynamic weight decay in minimizing overfitting during later training stages.

- *Insight*: These observations demonstrate that dynamic weight decay significantly reduces the time required to achieve grokking, providing evidence for its superiority over static strategies, as supported by [8] [7, 8].

2) Generalization:

- *Observation*: Sparse embeddings and dense aggregated attention weights enable efficient learning and robust performance across diverse input patterns.

- *Insight*: This aligns with findings in [1], which propose that hybrid sparsity-dense architectures enhance task generalization by leveraging complementary representations [16,17].

V. Discussion

Our study demonstrates that the weight decay, changed gradually, works well for Grokking. By changing over time, we are controlling the network's weights that allows the network to learn better and not unnecessarily overfit the training data. We start training with stronger control and then reduce it over sets of epochs. This agrees with the existing research regarding the adjustment of controls during training. [3,8]. While the network embedding layers (WE and WU) self-naturally adopted sparse connections to facilitate understanding of the internal operations and preventing overfitting, the attention layer indeed favored dense connectivity to pool information from every part of the input. [1]. It appears that the best results can be obtained by mixing sparse and dense connectivity in different parts of the network. We have applied this approach to simple math problems, but believe it might be particularly useful for more complex real-world problems [14, 18]. Futher investigation may explain how well this approach scales to pretrained transformer models and assess its applica- tion to tasks of different complexities, informing broader training dynamics and optimization strategies [10, 11].

VI. Analysis

A. Significance of Research

This work pushes the boundaries of grokking by proposing dynamic weight decay as an efficient means of speeding up generalization with the preservation of model interpretability. Experimental results support the claim of adaptive regularization in the case of evolving training dynamics outperforming static methods [3], [8]. The observed sparsity in embedding layers supports findings from [9], amplifies sparsity-driven interpretability for improvement in task-oriented performance. In particular, this work challenges the assumption of uniform sparsity across layers [10], demonstrating the efficacy of hybrid architectures with sparse embeddings and dense attention weights.

B. Broader Implications

Hybrid sparsity-dense architectures and dynamic weight decay are a scalable framework for achieving both interpretability and robust generalization. These results are of practical relevance for tasks that balance memorization and generalization; they provide the potential to scale up to large architectures. Dynamic weight decay overcomes limitations from static strategies and allows flexible optimization of training.

C. Comparative Analysis

This research aligns with prior work on the importance of weight decay for grokking [1], [2], while extending these insights by demonstrating the advantages of dynamic strategies for faster convergence. By combining sparsity and density analyses, it bridges gaps identified in [5], [11] and complements weight norm-based explanations for delayed generalization from [7]. This comprehensive approach underscores the interplay between sparsity, density, and adaptive regularization, providing a nuanced perspective on training dynamics.

VII. Conclusion

This work shows dynamic weight decay to be a method that aids grokking, having lower convergence time yet maintaining both interpretability and performance. It outperforms static weight decay because it adjusts regularization strength and potentially demonstrates scalability and flexibility in optimizing Model training.[9, 18].

The findings reveal a hybrid architecture where sparse embeddings (W_E, W_U) enhance task-specific generalization, and dense aggregated attention weights ensure robust feature extraction. This challenges uniform sparsity assumptions and highlights the complementary roles of sparsity and density in transformers [13, 17].

Dynamic weight decay bridges gaps in grokking research by uniting theoretical insights with practical performance gains. Its application opens the way toward scalable, adaptive strategies for complex tasks and further cements its status as a cornerstone for future deep learning optimization.

References

[1] D. Liu and S. Ritter, "Omnigrok: Grokking beyond algorithmic data," *arXiv*, vol. 2210.01117, 2023.

[2] A. Power, S. Abnar, and I. Sutskever, "Grokking: Generalization beyond memorization in neural networks," *arXiv*, vol. 2103.00020, 2022.

[3] A. Kosson, B. Messmer, and M. Jaggi, "Rotational equilibrium: How weight decay balances learning across neural networks," *arXiv*, vol. 2305.17212, 2023.

[4] N. Nanda, L. Chan, and T. Lieberum, "Progress measures for grokking via mechanistic interpretability," in *ICLR*, 2023.

[5] A. Pearce *et al.*, "Do machine learning models memorize or generalize?" *arXiv*, vol. 2304.06115, 2023.

[6] W. Merrill *et al.*, "Sparse subnetworks in grokking," *arXiv*, vol. 2303.11873, 2023.

[7] G. Minegishi, Y. Iwasawa, and Y. Matsuo, "Bridging lottery ticket and grokking: Is weight norm sufficient to explain delayed generalization?" *arXiv*, vol. 2305.07951, 2023.

[8] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv*, vol. 1711.05101, 2019.

[9] Z. Liu, E. J. Michaud, and M. Tegmark, "Omnigrok: Grokking beyond algorithmic data," *arXiv*, vol. 2210.01117, 2023.

[10] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv*, vol. 1803.03635, 2019.

[11] W. Merrill, N. Tsilivis, and A. Shukla, "A tale of two circuits: Grokking as competition of sparse and dense subnetworks," *arXiv*, vol. 2303.11873, 2023.

[12] V. Varma, R. Shah, and Z. Kenton, "Explaining grokking through circuit efficiency," *arXiv*, vol. 2309.02390, 2023.

[13] L. Wu *et al.*, "Dynamic weight decay in language models," *arXiv*, vol. 2010.08111, 2020.

[14] E. Zangrando, P. Deidda, S. Brugiapaglia, N. Guglielmi, and F. Tudisco, "Neural rank collapse: Weight decay and small within-class variability yield low-rank bias," *arXiv*, vol. 2402.03991, 2024.

[15] Z. Li, C. You, S. Bhojanapalli, D. Li, A. S. Rawat, S. J. Reddi, K. Ye, F. Chern, F. Yu, R. Guo, and S. Kumar, "The lazy neuron phenomenon: On emergence of activation sparsity in transformers," *arXiv*, vol. 2210.06313, 2022.

[16] B. Yoon, Y. Han, and G. E. Moon, "Spion: Layer-wise sparse training of transformer via convolutional flood filling," *arXiv*, vol. 2309.12578, 2023.