Progress Measures for Grokking on Real-world Tasks

Satvik Golechha ZSATVIK@GMAIL.COM
Independent Researcher

Abstract

Grokking, a phenomenon where machine learning models generalize long after overfitting, has been primarily observed and studied in algorithmic tasks. This paper explores grokking in real-world datasets using deep neural networks for classification under the cross-entropy loss. We challenge the prevalent hypothesis that the L_2 norm of weights is the primary cause of grokking by demonstrating that grokking can occur outside the expected range of weight norms. To better understand grokking, we introduce three new progress measures: activation sparsity, absolute weight entropy, and approximate local circuit complexity. These measures are conceptually related to generalization and demonstrate a stronger correlation with grokking in real-world datasets compared to weight norms. Our findings suggest that while weight norms might usually correlate with grokking and our progress measures, they are not causative, and our proposed measures provide a better understanding of the dynamics of grokking.

1. Introduction

Machine learning models are typically trained and evaluated using different datasets. Grokking [19] is a phenomenon where models generalize to low test-loss long after they overfit their training data. Grokking was first observed in algorithmic tasks [19], and several works have since attempted to understand why and how grokking happens in these algorithmic tasks [13, 17, 22].

Liu et al. [14] show that grokking can be observed in a "goldilocks zone" of weight norms on deep networks trained on non-algorithmic tasks. We follow their setup and consider the simple problem of multi-class classification using neural networks trained with the cross-entropy loss. The MNIST dataset [3] contains image inputs X, a set of labels Y, and a training dataset, $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_d, y_d)\}$.

For any arbitrary model acting as a classifier $h : X \times Y \to \mathbb{R}$, the softmax cross-entropy loss is given by:

$$L_{\text{CE}}(h) = -\frac{1}{D} \sum_{(x,y)\in D} \log \frac{\exp(h(x,y))}{\sum_{y'\in Y} \exp(h(x,y'))}.$$

The output of our model are the logits $o_h(x)$, which can give the class-wise output probabilities as $\text{softmax}(o_h^y(x)) \triangleq h(x,y)$. The set of all the parameters of the model (typically a neural network) is denoted by θ and the L_2 norm of the weights is denoted by $||\theta||_2^2$.

Our main contributions are as follows:

- We contradict the hypothesis of Liu et al. [14] that weight norms explain grokking by showing that generalization can occur way outside the "goldilocks zone" of low weight norm.

- We conceptually motivate three alternative progress measures for grokking on real-world data and show that they align well with generalization on a simple neural network trained on MNIST [3] and an LSTM-based model trained on the IMDb dataset [15].

2. Related Work

Grokking has been observed in both algorithmic [14, 19] and real-world tasks [9]. Several attempts have been made to explain grokking [13, 17, 22] in algorithmic tasks. Real-world data and models have been explored much less in terms of grokking, with Humayun et al. [9], Liu et al. [14] showing the phenomenon to occur in the first place, and attempting to explain them using some heuristics.

Progress measures [2] have been a powerful tool to understand deep networks by analyzing their training dynamics. Liu et al. [14], Nanda et al. [17], Varma et al. [22] introduce various measures to explain grokking on mathematical tasks, and Liu et al. [14] introduce the L_2 weight norm as the cause of grokking on real-world tasks. We show that this prevalent metric does not explain grokking in real-world tasks, and introduce three conceptually motivated progress measures that do.

3. L_2 Norms of Weights do not Explain Grokking

Unlike algorithmic tasks, for which grokking has been observed and studied in detail [13, 17, 22], there has been limited progress made to elicit and understand grokking in neural network models trained on real-world datasets. This was first observed by Liu et al. [14], where they found that by training a model using weight decay and starting with a high weight initialization, grokking can be observed on real-world datasets (such as MNIST [3]).

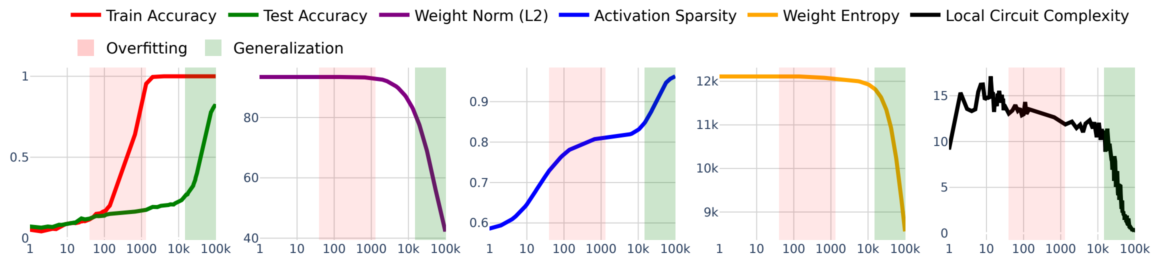

Figure 1: Grokking, as observed on MNIST by Liu et al. [14], with a reduction in the L_2 norm of the weights and the new progress measures introduced.

They hypothesized the weight norm to be the cause of grokking, which can occur in a range they call the "goldilocks zone". In this section, we refute this claim by showing that grokking can occur way outside this "goldilocks zone".

We employ a clever trick to elicit grokking while the L_2 norm of the weights increases during generalization. We keep the setup the same as in Liu et al. [14], and train a 3-layer, $200 - width$ MLP with ReLU non-linearities on the MNIST dataset using AdamW optimizer with weight decay. The hyperparameters are given in Appendix A. We see grokking and a reduction in the norm of the weights in the default setup. (see Fig. 1).

We then add the squared L_2 norm of the weights to the default cross-entropy loss function $\mathcal{L}$:

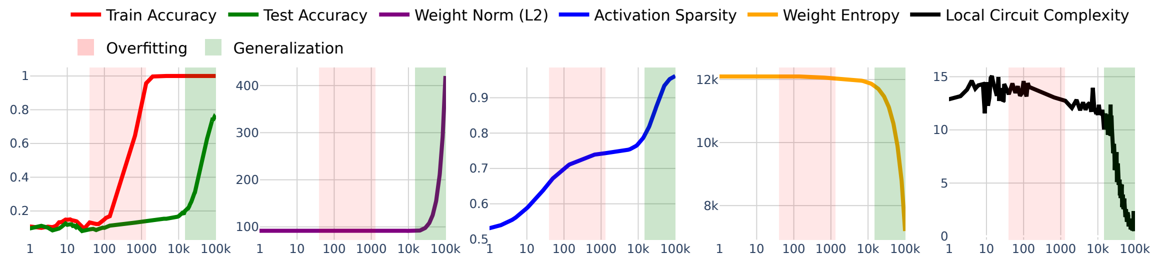

Figure 2: Loss landscape for the modified setup showing an increase in the L_2 norm of the weights and the new progress measures introduced.

$$\mathcal{L}' = \mathcal{L} - \delta \sum_{w \in \theta} |w|^2$$

This pushes the weights to increase their norm. A simple manual search shows grokking to occur at $\delta = 2e - 10$, while the weight norm keeps on increasing, as seen in Fig. 2. Note that for the AdamW optimizer, this is implemented differently than weight decay.

Thus, a high weight norm initialization and weight decay is not the cause for grokking as hypothesized by Liu et al. [14]. In the next section, we motivate three progress measures that conceptually align with existing notions of generalization, and show that these measures better capture the occurrence of grokking in real-world datasets.

4. Progress Measures for Grokking on Real-world Tasks

4.1. Activation Sparsity

Let $\mathbf{a}_i$ be the activations of a given layer for input x_i. The sparsity S is defined as the fraction of activations that are below a threshold τ.

$$S = \frac{1}{D} \sum_{i=1}^{D} \frac{1}{n} \sum_{j=1}^{n} \mathbb{I}(a_{i,j} < \tau)$$

where D is the number of training examples, n is the number of activations in the layer, $\mathbb{I}(\cdot)$ is a condition indicator function, and $a_{i,j}$ is the j-th activation for the i-th input.

Activation sparsity measures how many activations are near zero, indicating how many neurons are effectively turned "off", with a higher sparsity implying that fewer neurons are actively contributing to the output.

Several prior works (such as [8, 12, 18]) study the properties of activation sparsity in the context of neural networks. Specifically, Li et al. [12] find sparse MLP layers in networks that generalize well, Merrill et al. [16] develop a measure of *effective sparsity* to explain grokking on an algorithmic task, and Huesmann et al. [8] observe an increase in activation sparsity right before an increasing test loss. In line with these observations, we observe activation sparsity to plateau right before the model starts to grok (see Figs. 1 and 2).

4.2. Absolute Weight Entropy

Let $\mathbf{W}$ be the weight matrix of a layer in a neural network with shape (m, n). The Shannon entropy H [21] of the absolute weights is defined as:

$$H(\mathbf{W}) = -\sum_{i=1}^{m}\sum_{j=1}^{n} |w_{ij}| \log |w_{ij}|$$

While the weights of a network do not form a probability distribution, $abs(w_{ij})$ can be intuitively thought of as a measure of the impact of feature i in layer N affecting feature j in layer $N+1$, where features are represented by neurons in the standard basis. While some model components represent features in superposition (such as a transformer's residual stream [4], MLP layers do have a privileged basis due to non-linearities [4]. Thus, the absolute weight entropy captures the spread and magnitude of these weights.

During training, the optimization process adjusts the weights to minimize the cross-entropy loss. Let's analyze how this adjustment impacts the absolute weight entropy.

4.2.1. INITIALIZATION

Typically, weights are initialized using schemes such as the Glorot uniform initialization [11], which distributes weights uniformly over an interval $[-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}}]$. This initialization is designed to keep the scale of gradients roughly consistent across layers. With $w_{ij} \sim U(a, a)$, where $a = \sqrt{\frac{6}{m+n}}$, the initial weight entropy is given by $H(\mathbf{W}) = mn\left(\frac{a}{4} - \frac{a}{2}\log a\right)$. (See Appendix B for a proof of the same).

4.2.2. DURING TRAINING

During training, gradient descent iteratively updates the weights to reduce the cross-entropy loss $\mathcal{L}$. These updates are driven by the need to fit the training data better, which often leads to some weights increasing in magnitude (if they are critical for reducing the loss) and others decreasing or becoming very small (if they are less critical).

Critical Weights Increase: Weights that contribute significantly to reducing the loss may increase in magnitude. However, due to the lottery ticket hypothesis (shown to hold to a good degree for MLPs trained on MNIST [5]), this is often limited to a much smaller subset of the weights. Since $|w_{ij}| \log |w_{ij}|$ increases with $|w_{ij}|$, these weights contribute more to $H(W)$.

Non-Critical Weights Decrease: Many weights may decrease in magnitude as the model learns to focus on the most critical features. For weights close to zero, $|w_{ij}| \log |w_{ij}|$ approaches zero, contributing less to $H(W)$.

Given these dynamics, one can expect that while a few weights might increase in magnitude, the overall trend for the majority of weights is towards smaller magnitudes, leading to $H(\mathbf{W}_{\text{trained}}) < H(\mathbf{W}_{\text{init}})$.

This reduction in entropy indicates a more concentrated weight distribution, which contributes to the model's robustness and generalization ability. Weight entropy measures the uncertainty or spread of the weight distribution, and a lower entropy is indicative of a compact and concentrated weight distribution, a sign of robust generalization.

We see in Figs. 1 and 2 that the weight entropy sharply decreases with generalization in both increasing and decreasing weight norms and does not change during overfitting, verifying our hypothesis. Several prior works have studied different forms of entropy to regularize and stabilize neural network training [1, 6, 25] and to decrease their effective depth [20, 24].

4.3. Approximate Local Circuit Complexity

Past works to calculate the effective circuit complexity of a neural network have been both computationally expensive and difficult to scale [9, 10]. [9] create spline partitions on the network's input space to show that deep networks grok on adversarial datapoints. Here, we come up with a simple and scalable progress measure which we call the approximate local circuit complexity that is another meaningful measure of grokking (see Figs. 1 and 2).

Let $\mathbf{o}_h^{(\mathbf{W})}(x)$ be the output logits of a model for input x with $\mathbf{W}$ being the weights of a layer of choice, and let $\mathbf{o}_h^{(\mathbf{W}')}(x)$ be the logits when 10% of the model's weights are made zero, resulting in a new weight matrix $\mathbf{W}'$. Intuitively, this corresponds to turning off a random 10% of the layer's computation in a similar fashion as dropout during training [7]. The progress measure is then defined as the KL divergence D_{KL} between the two logit distributions:

$$D_{\text{KL}}\left(P(\mathbf{o}_h^{(\mathbf{W})}\|\mathbf{x}) \parallel P(\mathbf{o}_h^{(\mathbf{W}')}\|\mathbf{x})\right) = \sum_{i=1}^{D}\sum_{y\in Y} P(\mathbf{o}_h^{(\mathbf{W})}(x_i,y)) \log\left(\frac{P(\mathbf{o}_h^{(\mathbf{W})}(x_i,y))}{P(\mathbf{o}_h^{(\mathbf{W}')}(x_i,y))}\right)$$

where $P(\mathbf{o}_h^{(\mathbf{W})}(x_i, y))$ is the softmax probability distribution of logits under the original weights $\mathbf{W}$, and $P(\mathbf{o}_h^{(\mathbf{W}')}(x_i, y))$ is the same under perturbed weights $\mathbf{W}'$.

Conceptually, approximate local circuit complexity measures the sensitivity of the model's output to small perturbations in the weights. A lower KL divergence indicates that the model's logits are more robust to these perturbations, suggesting that the model has learned more stable and reliable internal representations, thereby leading to generalization. We empirically verify this hypothesis in Fig. 1. We also show in Appendix C that the same model trained with dropout does not show a delay in generalization.

Thus, we show that a decrease in the L_2 norm of the weights is not necessarily correlated with generalization as found by [14], and clearly doesn't imply a causation as claimed by their "LU mechanism". We show limited results on the IMDb dataset [15] in Appendix D.

5. Conclusion

We introduce three conceptually motivated progress measures for grokking on real-world data and show that the correlation of grokking with a decrease in the L_2 weight norm is not required, and thus prove that weight norms are not the causal explanation behind grokking. We show generalization to occur at high weight norms while our measures still work well. As potential future directions, it would be interesting to see how the performance dynamics changes when we make these progress measure differentiable using approximations and add them to the loss function to control grokking. Another interesting direction would be to build a theoretical framework for grokking based on these measures and to concretely ascertain their occurrence during generalization. Lastly, it would be great to look at other real-world datasets to see the generalizability of the results.

References

[1] Vladimir Araujo, Julio Hurtado, Alvaro Soto, and Marie-Francine Moens. Entropy-based stability-plasticity for lifelong learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3721–3728, 2022.

[2] Boaz Barak, Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: Sgd learns parities near the computational limit. *Advances in Neural Information Processing Systems*, 35:21750–21764, 2022.

[3] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[4] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1:1, 2021.

[5] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[6] Marylou Gabrié, Andre Manoel, Clément Luneau, Nicolas Macris, Florent Krzakala, Lenka Zdeborová, et al. Entropy and mutual information in models of deep neural networks. *Advances in neural information processing systems*, 31, 2018.

[7] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[8] Karim Huesmann, Luis Garcia Rodriguez, Lars Linsen, and Benjamin Risse. The impact of activation sparsity on overfitting in convolutional neural networks. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part III*, pages 130–145. Springer, 2021.

[9] Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. Deep networks always grok and here is why. *arXiv preprint arXiv:2402.15555*, 2024.

[10] Romuald A Janik and Przemek Witaszczyk. Complexity for deep neural networks and other characteristics of deep feature representations. *arXiv preprint arXiv:2006.04791*, 2020.

[11] Siddharth Krishna Kumar. On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863*, 2017.

[12] Zonglin Li, Chong You, Srinadh Bhojanapalli, Daliang Li, Ankit Singh Rawat, Sashank J Reddi, Ke Ye, Felix Chern, Felix Yu, Ruiqi Guo, et al. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. *arXiv preprint arXiv:2210.06313*, 2022.

[13] 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.

[14] Ziming Liu, Eric J Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data. In *The Eleventh International Conference on Learning Representations*, 2022.

[15] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[16] William Merrill, Nikolaos Tsilivis, and Aman Shukla. A tale of two circuits: Grokking as competition of sparse and dense subnetworks. *arXiv preprint arXiv:2303.11873*, 2023.

[17] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

[18] Ze Peng, Lei Qi, Yinghuan Shi, and Yang Gao. Theoretical explanation of activation sparsity through flat minima and adversarial robustness. *arXiv preprint arXiv:2309.03004*, 2023.

[19] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

[20] Victor Quétu, Zhu Liao, and Enzo Tartaglione. The simpler the better: An entropy-based importance metric to reduce neural networks' depth. *arXiv preprint arXiv:2404.18949*, 2024.

[21] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.

[22] Vikrant Varma, Rohin Shah, Zachary Kenton, János Kramár, and Ramana Kumar. Explaining grokking through circuit efficiency. *arXiv preprint arXiv:2309.02390*, 2023.

[23] Christian Bakke Vennerød, Adrian Kjærran, and Erling Stray Bugge. Long short-term memory rnn. *arXiv preprint arXiv:2105.06756*, 2021.

[24] Simon Wiedemann, Arturo Marban, Klaus-Robert Müller, and Wojciech Samek. Entropy-constrained training of deep neural networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[25] Wanbing Zou, Song Cheng, Luyuan Wang, Guanyu Fu, Delong Shang, Yumei Zhou, and Yi Zhan. Increasing information entropy of both weights and activations for the binary neural networks. *Electronics*, 10(16):1943, 2021.

Appendix A. Hyperparameters

Hyperparameter	Value
Train Points	1000
Test Points	1000
Optimization Steps	100000
Batch Size	1000
Loss Function	MSELoss
Optimizer	AdamW
Weight Decay	0.01
Learning Rate	1e-3
Initialization Scale	8.0
Depth	3
Width	200
Activation	ReLU

Table 1: Hyperparameters used in the experiment. We train a 3-layer, 200-width MLP with ReLU non-linearities on the MNIST dataset using the AdamW optimizer with weight decay.

Appendix B. Closed-form Weight Entropy for Uniform Initialization

Assume the weights are initialized uniformly over an interval $[-a, a]$. For a uniform distribution, each weight w_{ij} has an equal probability of taking any value within this interval.

The closed-form for the weight entropy H can be derived as follows:

$$w_{ij} \sim U(-a, a)$$

The weight log-entropy H is given by:

$$H(\mathbf{W}) = -\sum_{i=1}^{m}\sum_{j=1}^{n} |w_{ij}| \log |w_{ij}|$$

For uniform initialization (such as the commonly used Xavier initialization scheme [11], let's assume that the weights are evenly distributed across the interval $[-a, a]$. We can approximate the sum by integrating over the interval for a continuous approximation:

$$H(\mathbf{W}) \approx -mn \int_{-a}^{a} |w| \log |w| \cdot \frac{1}{2a}\, dw$$

Here, $\frac{1}{2a}$ is the probability density function of the uniform distribution over $[-a, a]$. Splitting the integral into two parts (from $-a$ to 0 and from 0 to a):

$$H(\mathbf{W}) \approx -mn \left(\int_{-a}^{0} |w| \log |w| \cdot \frac{1}{2a}\, dw + \int_{0}^{a} |w| \log |w| \cdot \frac{1}{2a}\, dw \right)$$

Since $|w| = -w$ for $w < 0$ and $|w| = w$ for $w > 0$, we have:

$$H(\mathbf{W}) \approx -mn \left(\int_0^a w \log w \cdot \frac{1}{a} \, dw \right)$$

This simplifies to:

$$H(\mathbf{W}) \approx -mn \cdot \frac{1}{a} \int_0^a w \log w \, dw$$

To solve the integral, use integration by parts with $u = \log w$ and $dv = w \, dw$:

$$\int_0^a w \log w \, dw = \left[\frac{w^2}{2} \log w \right]_0^a - \int_0^a \frac{w^2}{2} \cdot \frac{1}{w} \, dw = \left[\frac{w^2}{2} \log w \right]_0^a - \frac{1}{2} \int_0^a w \, dw$$

The second integral is straightforward:

$$\int_0^a w \, dw = \left[\frac{w^2}{2} \right]_0^a = \frac{a^2}{2}$$

Combining the results:

$$\left[\frac{w^2}{2} \log w \right]_0^a = \frac{a^2}{2} \log a - (0 \cdot \log 0 \text{ (which is 0)})$$

$$\int_0^a w \log w \, dw = \frac{a^2}{2} \log a - \frac{1}{2} \cdot \frac{a^2}{2} = \frac{a^2}{2} \log a - \frac{a^2}{4}$$

Therefore, we get the following closed-form expression for H:

$$H(\mathbf{W}) \approx -mn \cdot \frac{1}{a} \left(\frac{a^2}{2} \log a - \frac{a^2}{4} \right) = -mn \left(\frac{a}{2} \log a - \frac{a}{4} \right) = mn \left(\frac{a}{4} - \frac{a}{2} \log a \right)$$

Appendix C. Effects of Dropout

We keep the rest of the setup the same as in Sec. 4, but add dropout [7] to each linear layer of our neural network. We set the dropout fraction as $p = 0.3$.

We see in Fig. 3 that the addition of dropout makes delayed generalization (grokking) completely vanish. This is in line with our experiments with the circuit complexity progress measure (see Sec. 4).

Appendix D. Progress Measures on the IMDb Dataset

We show our progress measures on an LSTM-based [23] model trained on the IMDb Movie Review Dataset [15], with the exact same setup and hyperparameters as in [14]. We observe in Fig. 4 that even with a different model architecture, our progress measures can be effectively used to study and predict generalization.

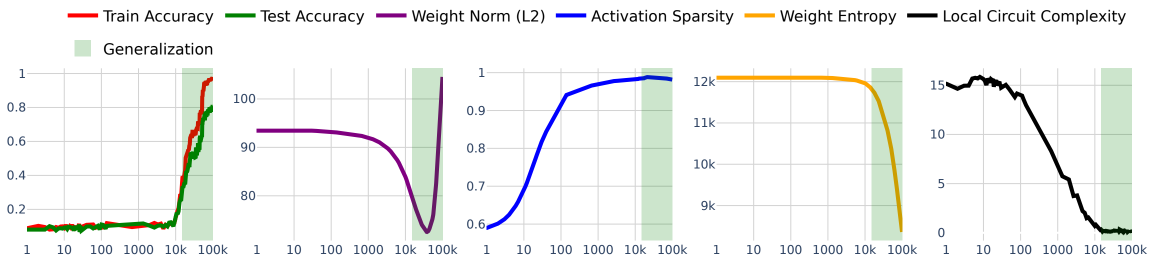

Figure 3: The effects of dropout on delayed generalization and the progress measures. Note the absence of an overfitting zone as the model does not exhibit grokking.

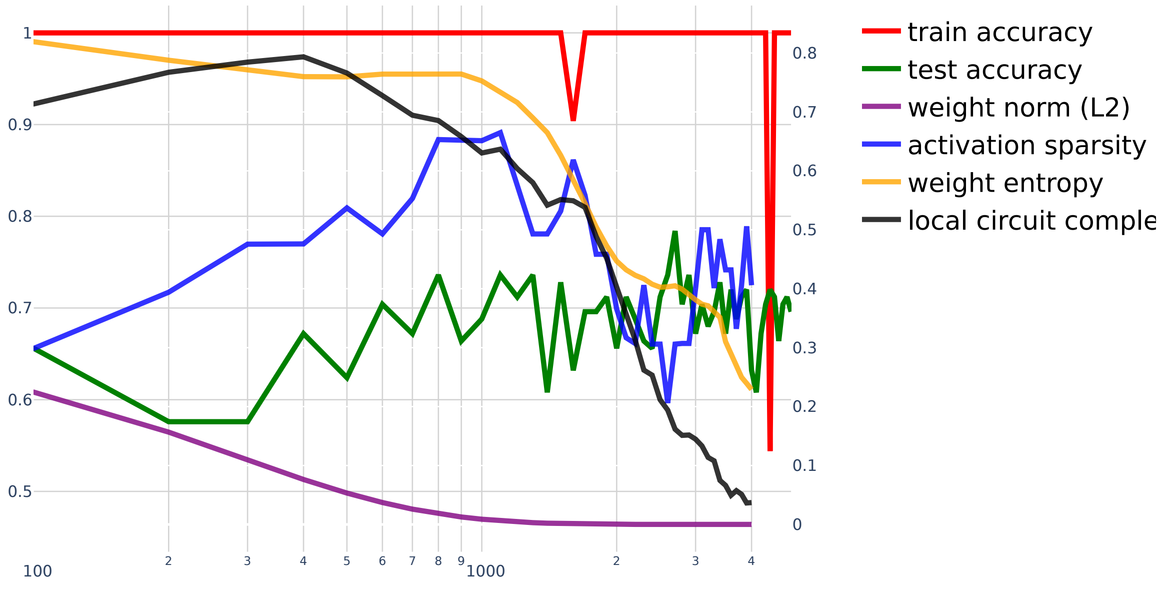

Figure 4: Grokking, as observed on the IMDb dataset by [14], with a reduction in the L_2 norm of the weights and the new progress measures introduced.