Investigating Training Dynamics of Transformer Models on Algorithmically Generated In-Context Learning Datasets

Harry Zhao

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-96
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-96.html

May 11, 2023

Acknowledgement

I would like to thank my research mentor, Ruiqi Zhong, for his continued support and mentorship
throughout my undergraduate and graduate research projects. This work would not have been
possible without his guidance. I would also like to thank Professor Dan Klein and Professor Jacob
Steinhardt for being on my master's committee and providing their valuable feedback on this
project.

Investigating Training Dynamics of Transformer Models on Algorithmically Generated In-Context Learning Datasets

Harry Zhao

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

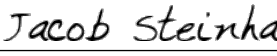
Committee

Dan Klein
Research Advisor

05/10/2023

(Date)

★ ★ ★ ★ ★ ★ ★



Jacob Steinhardt
Second Reader

05/10/2023

(Date)

Abstract

We aimed to study the training dynamics and the internal interpretability of transformer models by formulating an algorithmically generated in-context learning task and training small models that can learn to generalize the task with 100% test accuracy. We found clear indications of phase change behavior that are indicative of emergent abilities, invariant attention patterns across different one-attention-head models, and early determination during training of convergence probability. We found promising future work directions for further studying transformer models, both small models and generalizations on larger models.

Acknowledgements

I would like to thank my research mentor, Ruiqi Zhong, for his continued support and mentorship throughout my undergraduate and graduate research projects. This work would not have been possible without his guidance. I would also like to thank Professor Dan Klein and Professor Jacob Steinhardt for being on my master's committee and providing their valuable feedback on this project.

Contents

1 Introduction

The transformer model was introduced in 2017 by Vaswani et al.[10] as a new architecture for machine translation. Since then, it has revolutionized natural language processing with its new paradigm for sequence modeling. These models have been widely adopted due to their ability to capture long-range dependencies and avoiding long time-chain backpropagations. As transformer models become more prevalent in NLP, their interpretability becomes a crucial issue. Interpretability refers to the ability to understand how a model makes its predictions, as well as understanding how the model is trained to achieve the performance that it has. With the advent of large language models (LLMs) such as OpenAI's GPT series (Brown et al., 2020 [4]), the understanding of how the models train and develop becomes increasingly obscure.

Previous work have been done to study the training dynamics of transformer models. Wei et al., 2022[11] have studied emergent phenomenon of neural networks. They have found that new abilities, that is, abilities not present in smaller models at all, emerge as training data and model size scale up. They cannot be predicted by extrapolating performance of smaller models; a qualitative change happens at a larger scale. Barak et al., 2023 [3] took the emergent phenomenon as motivation, and worked on hidden progress measures, that is, measures that are not test/training accuracy, to explore the progress of neural network training for such phenomenon. There have been doubts regarding the phenomenon of emergent abilities. Schaeffer et al. 2023 [9] claimed that emergent abilities are demonstrated through the metric of choice of the researcher instead of being inherent in the model learning process; they attempted to argue that different metrics will either show or ablate an emergent ability. Some sections of this work attempt to refute these claims.

Akyürek et al. 2022[1] claimed that large language models (LLMs) exhibit in-context learning, that is, constructing new predictors from labeled examples represented in the input. They have demonstrated that transformer models, without being given the explicit formulaic guidance, can converge to the optimal closed-form linear models in the in-context learning task, and such in-context learners share algorithmic features with the classic predictors. Garg et al. 2023[6] similarly demonstrated the transformer model's ability to learn complex function classes such as sparse linear functions, two-layer neural networks, and decision trees, in-context. Chen et al. 2023[5] investigated fine-tuning large language models with in-context learning with promising results, which will prove soon beneficial with the current emergence of large language models.

Power et al., 2022[8] studied the generalization of neural networks on small algorithmically generated datasets, from their work we take much inspiration from. Specifically they have identified a pattern of "grokking" in the data, where the neural network improves performance to perfect generalization. This improvement in generalization sometimes can happen well past the point of overfitting. They also studied generalization and its relationship with dataset sizes, and found unsurprisingly that smaller datasets required increasing amounts of optimization for generalization. Nanda et al., 2023[7] proposed mechanistic explanations of reverse engineering the mechanisms of the transformer network to study generalization behaviors; specifically, they were able to identify that the transformer network they trained was using fourier transform methods to compute modular arithmetic. It is from the foundation of these results that we begin our work.

2 Background

2.1 The Transformer Model

The transformer model, proposed by Vaswani et al., 2017[10], are originally neural network models that map from a sequence of input tokens $\mathbf{x} = [x_1, x_2, ..., x_n]$ to a sequence of output tokens $\mathbf{y} = [y_1, y_2, ..., y_n]$. This is achieved with an encoder-decoder architecture. In this work we are solely interested in the encoder-only architecture for classification tasks. For each layer of the encoder, we first compute the self-attention:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{.1}$$

With multi-head attention, we concatenate outputs of several independent heads of the self-attention:

$$\begin{aligned} &MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \\ &\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{.2}$$

The matrix W^O projects the result of the multi-head attention down to the single head dimension. The Q, K, V matrices in our implementation are all the input matrix $\mathbf{X}$ first fed through a layernorm (Ba et al., 2016 [2]):

$$layernorm(\mathbf{X}) = \frac{\mathbf{X} - \mathbb{E}(\mathbf{X})}{\sqrt{Var(\mathbf{X})}} \tag{.3}$$

We then add a skip-connection layer

$$\mathbf{X}_{int} = \mathbf{X} + MultiHead(layernorm(\mathbf{X})) \tag{.4}$$

and finally a feedfoward layer and another skip connection, to the output.

$$\mathbf{X}_{out} = Feedforward(\mathbf{X}_{int}) + \mathbf{X}_{int} \tag{.5}$$

where the $Feedforward$ layer is defined by a $layernorm$, a $Linear$ layer,

$$Linear(\mathbf{X}) = W^{d \times d}\mathbf{X} + \mathbf{b} \tag{.6}$$

where d is the dimension of the embeddings and $\mathbf{b}$ the bias vector independent for each $Linear$ layer, a $ReLU$ layer,

$$ReLU(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \tag{.7}$$

and another $Linear$ layer.

3 Experiment Setup

3.1 Algorithmically Generated In-Context Learning Task

We set up the task as a classification task. The input I into the model is a [BOS] token, a sequence S, a [SEP] token, and then the prediction prompt P followed by an [EOS] token. The sequence S is sampled according to the following:

$$S = (x_1, y_1, x_2, y_2, ...x_n, y_n) \tag{.8}$$

where $x_i \in \{1, 2, 3, ..., 26\}$ and $y_i \in \{A, B, C, ..., Z\}$. The sequence length $2n$ is variable and $n \leq 26$. We uniformly sample x_i and y_i without repetition. We then sample P according to the following:

$$P = x_k, k \sim Uniform(1, n) \tag{.9}$$

That is, P is uniformly sampled from one of x_i. If $P = x_k$, then the solution to the prompt that the model attempts to predict is y_k, the letter at position $2k$ in the sequence S. For example, the input

I= [BOS] 24 P 6 Y 8 Q 9 H 25 J 12 D 22 U 21 G 13 B 14 V 5 W 4 C 17 B 15 M 19 I [SEP] 9 [EOS]

has solution H, since $P = 9$, $k = 4$, and $y_4 = H$. As another example, the input

I= [BOS] 3 L 12 K 20 A 5 A 26 Y 11 G [SEP] 12 [EOS]

has solution K, since $P = 12$, $k = 2$, and $y_2 = K$.

3.2 Model

We use a simple transformer for this task, with separate embeddings for each of the tokens. We add to the input $X = embed(I) \in \mathbb{R}^{(2n+4)\times 512}$ sinusoidal positional embeddings and then use 2 layers of the transformer encoder layer. We use no dropout for any of these experiments. We then take the first column vector from the output matrix E of the transformer encoder, take a linear neural layer and $LogSoftmax$

$$out = LogSoftmax(E[0] \cdot W^{512\times 26} + \mathbf{b}) \tag{.10}$$

for the prediction result. We performed extensive experiments for the 2 layer 1 attention head per layer model (which we call the one-head model), and some experiments for the 2 layer 2 attention heads per layer model (which we call the two-head model).

3.3 Training

We train the model using synthetically generated data, and each training batch contain new datapoints drawn from the data distribution; since they are unlikely to overlap datapoints the model has seen before, we can interpret the training loss/accuracy as the evaluation loss/accuracy. We train the model with a maximum of 2 million steps, with a 1e-4 initial learning rate and a linear learning rate warmup scheduler. We use the AdamW optimizer. Training hyperparameters were not changed throughout the entirety of the experiments ran.

4 Results

We first wanted to see whether learning such a task with very few layers/heads on the transformer model was possible. After finding convergent models, we wanted to investigate how the model

computes its solution, through probing attention weights. We also wanted to see whether such attention patterns were emergent with model training. Then we wanted to find hidden progress measures for the model training progress that was not simply train/test accuracy. Finally we were fascinated by convergent probabilities after training the model for a number of steps, and investigated the relationship between convergence probability and training steps. The following sections detail these investigations:

- Convergence phase change behavior, the specific conditions under which the model converges to accuracy 1
- Attention pattern behavior, the common patterns that emerge in models that do achieve accuracy 1
- l_2 norm behavior, correlations between the model l_2 norm and accuracy
- Convergence probability behavior, where whether the model converges to accuracy 1 is determined early in training

4.1 Result 1: Convergence Phase Change Behavior

Introduction

We first wanted to investigate whether and how such simple models would be able to learn the task given. In addition to training models, we wanted to understand the dynamics during the course of training and how the model achieves the performance that it does.

Results

We first trained several 1 head models, and plotted their training curves:

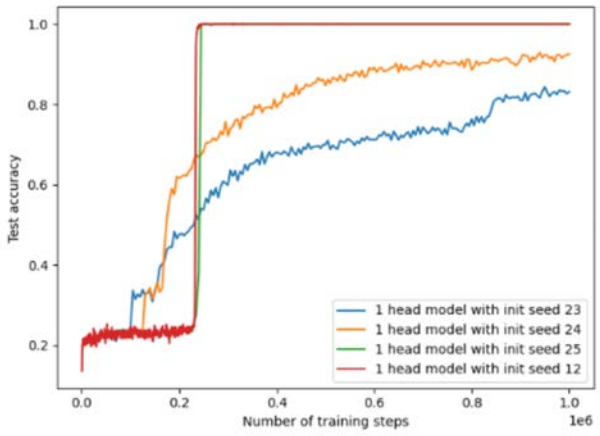


Figure 1: Examples of one-head model convergence curves, where we observe either a phase-change to accuracy 1 or a continual climb with limiting accuracy < 1.

We have number of training steps on the x axis and test accuracy on the y axis. We have discovered that for the 1head model, two types of behavior were possible under the set training parameters(across hundreds of training runs): either the test accuracy rapidly converges to 1 from 0.2 early on during training, or the test accuracy slowly approaches 0.9 (with some phase change behavior in the middle, for example jumping from 0.2 to 0.4 then to 0.6) range but never converges to 1.

The phase change behavior is confirmed by the training loss curve:

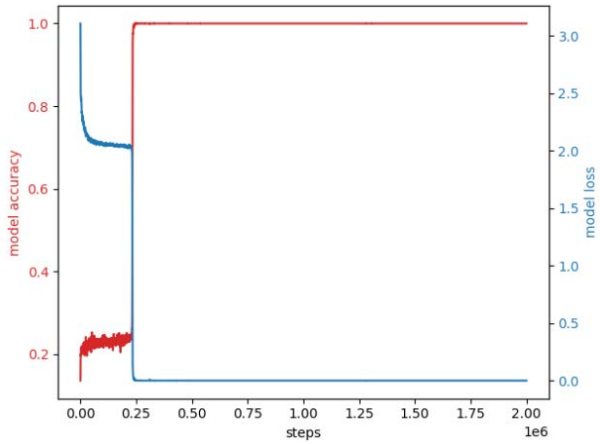


Figure 2: One-head model training loss vs accuracy curves, where the training loss decreases sharply as the model accuracy increases sharply.

We have the training steps on the x-axis, training loss/model accuracy on the y-axes.
We see that the 2head model behavior is a little bit more diversified, but the phase change behavior (accuracy jumping from 0.2 to 1) is still apparent and prominent.

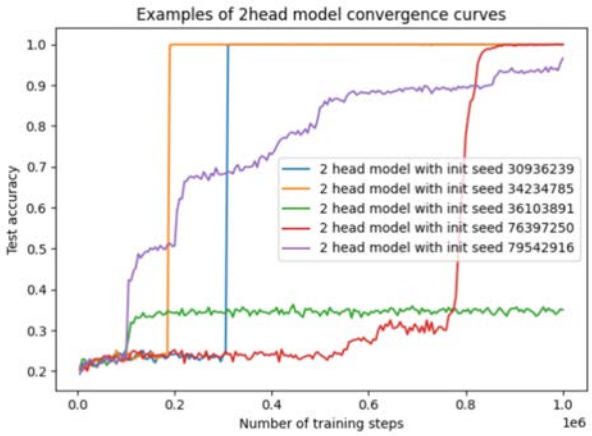


Figure 3: Examples of two-head model convergence curves, where phase change behavior is still prominent

Observations

We can conclude that the phase change behavior/emergent abilities across models are prominent. With the one-head model we have concluded that the phase change behavior is the only possibility through which the model achieves generalization. We note that the metrics and indications of such behavior are simple test accuracy/training loss, and are not results of overengineered metrics deliberately chosen, as Schaeffer et al., 2023[9] would suggest.

4.2 Result 2: Attention Pattern Emergence

Introduction

We wanted to see how the models arrive at the solution internally and conducted experiments with the attention patterns/weights. The attention weights is the matrix

$$AttentionWeights(Q,K) = softmax(\frac{QK^T}{\sqrt{d_k}}) \tag{.11}$$

As part of the step within

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{.12}$$

These weights indicate for each of the tokens, which of the other tokens the model *attends* to. We wanted to also evaluate systematically how attention patterns generalize across different data points, and we performed some predictions on the attention pattern. Specifically, given an attention weights matrix $W \in \mathbb{R}^{I \times I}$ where I is the length of the input, we define the *maximally attended tokens*, T of the sequence as

$$T = Argmax(W, dim = -1) \tag{.13}$$

where T is a vector of length I, with an integer indicating the position for each of its elements.

Results

We found that for the one-head model, several converged models exhibit very similar attention behavior:

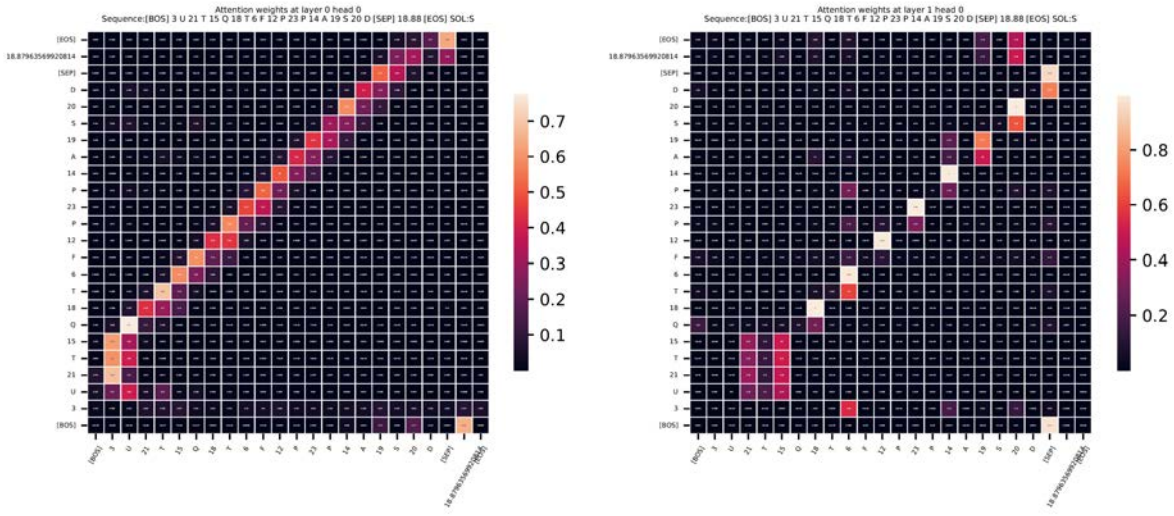


Figure 4: One-head model attention patterns, where we see clear indications of systematic attention behavior

We define $Pred_i^l$ as the prediction of the ith position of the ground-truth maximally attended tokens vector T at layer l. For the head of the first layer, predicting that the maximally attended token lies within two tokens on the offset diagonal, with exceptions of the beginning and end tokens, that is

$$Pred_i^1 \in \{i-3, i-4\} \tag{.14}$$

gives 97% accuracy. For the second layer, predicting that the first token attends to 3 tokens to the right of the solution token, that is, if k is the position of the solution token then

$$Pred_1^2 = k+3 \tag{.15}$$

and every other token attends to itself on the diagonal, that is,

$$Pred_i^2 = i, i \in \{7, 9, 11, 13, 15, 17, 19, 21\} \tag{.16}$$

also gives 97% accuracy. The same prediction hard code can be ran on two models trained on different random seeds, and the prediction accuracy remains the same.

We found that model convergence is highly correlated with the attention pattern emergence. That is, as the model accuracy phase changed, so did the attention pattern prediction accuracy.

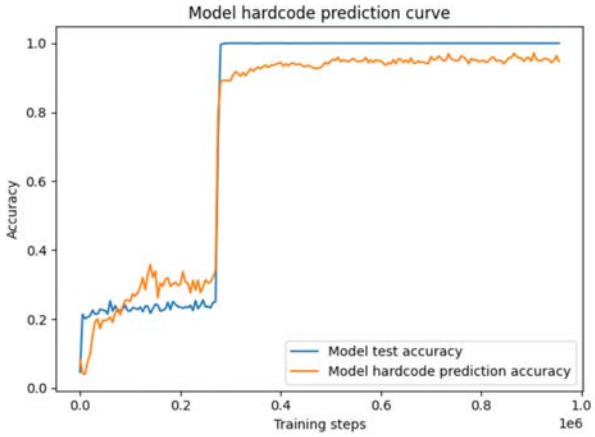


Figure 5: One-head model attention prediction accuracy vs task performance accuracy curve, where generalization happens exactly as the attention pattern emerges

Observations

We have also found no attention pattern before the test accuracy phase change. We can conclude that the test accuracy emergence is highly correlated with the attention pattern emergence, and perhaps, model generalization is achieved *through* finding the attention weights.

4.3 Result 3: Model Convergence vs L2 Norm

Introduction

We wanted to find indicators of model development progress during and after model training, and inspired by Barak et. al 2022, we measured the l_2 norm of the model as it trained. We define the l_2 norm of the model M as

$$\|M\|_2 = \sum_p p^2, p \in Parameters(M) \tag{.17}$$

Results

For several training runs, we investigate the l_2 norm of the model as they trained.

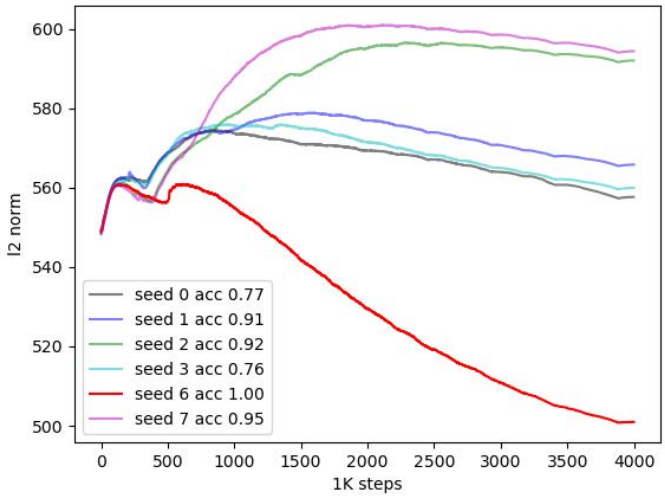


Figure 6: l_2 norm of the models with final indicated accuracies, where the model finding the exact solution will have the smallest l_2 norm

We have the l_2 norm on the y-axis and the number of training steps on the x-axis. We can see that the model finding the exact solution will have the smallest l2 norm.

We measured the model l_2 norm at the end of the training, and plotted it against the accuracy of the model:

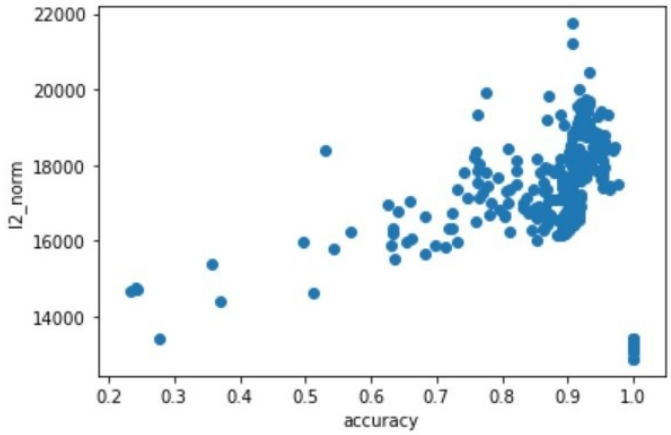


Figure 7: l_2 norm of the models vs model final accuracies, where the models finding the exact solution will have small l_2 norms but the models that do not find the exact solution will exhibit a positive l_2 norm-accuracy correlation

We observe that when the model finds the exact solution, the l2 norm is small; when the model does not find the exact solution, the norm is positively correlated with accuracy.

Observations

We can conclude that for this particular task, the l_2 norm serves as a progress measure of the model's training. From the final plot we find positive correlations between accuracy and norm when the model does not find the exact solution, and the model norm being small when it does find the exact solution. We may conjecture that when the model does not find the exact solution, it attempts to approximate an answer using increasingly complex and variant behavior.

4.4 Result 4: Convergence Probability Determinations

Introduction

We first found that for the same randomly initialized model M_i, whether the model will converge to accuracy 1 changes as data generation seeds change.

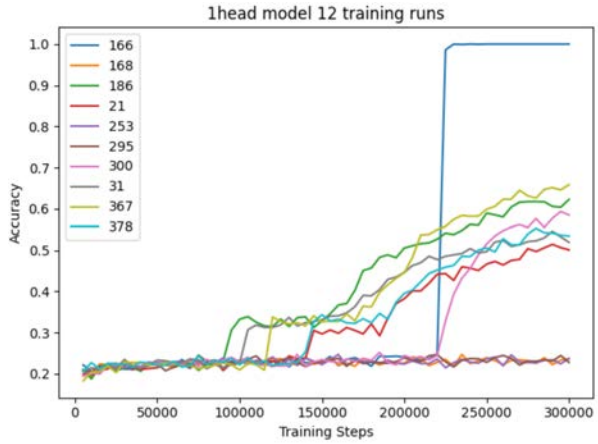


Figure 8: Training runs of the same one-head model under different data generation seeds, where the data provided to the model training impacts whether it will converge to accuracy 1

In fact, when we train the same initialized one-head model under different data generation seeds, the probability of convergence is around 0.1. We wanted to measure, when, if at all, would the model decisively converge, no matter which data generation seed is being used.
We define

$$\mathbb{S}(M_i^j) = \text{Training step at which model } M_i \text{converges to accuracy 1, training with seed } j \quad (.18)$$

Conveniently, we use M_i^j to denote M_i being trained under seed j. Then we define

$$\mathbb{P}(M_i^j, t) = \mathbb{P}[\mathbb{S}(M_i^j) < \infty] \text{ after training with seed } j \text{ for } t \text{ steps} \quad (.19)$$

That is, $\mathbb{P}(M_i^j, t)$ is the probability that model M_i will converge to accuracy 1, after being trained for t steps with training seed j. For concrete experiments, We first find data generation seed j under which model M_i converges to accuracy 1. Then we trained model M_i under seed j for t steps, saved the model, and then trained it under different training seeds to approximate $\mathbb{P}(M_i^j, t)$. We substitute 300, 000 for ∞ in the above for empirical experiments, since we have found that

empirically, if $\mathbb{S}(M_i^j) < \infty$, then $\mathbb{S}(M_i^j) \in [200000, 280000]$.

Assume that $\mathbb{S}(M_i^j) = 250,000$, we hypothesize

$$\lim_{t \to 250,000} \mathbb{P}(M_i^j, t) \to 1 \tag{.20}$$

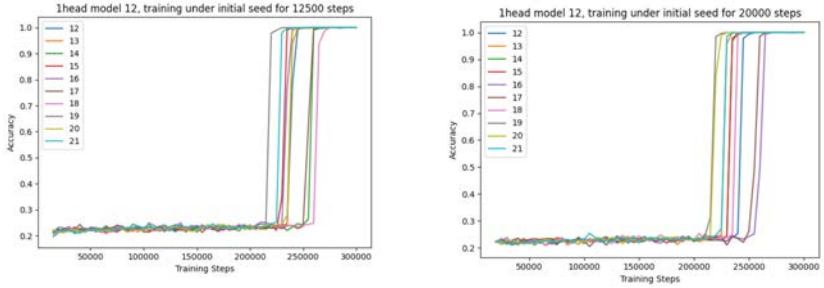
That is, if M_i^j converges to accuracy 1 at training step $t = 250,000$, varying the training seed at step $t = 249,999$ would not affect its convergent behavior.

Results

We first found that surprisingly, for model M_{12}^{11},

$$\lim_{t \to 12500} \mathbb{P}(M_{12}^{11}, t) \to 1 \tag{.21}$$

despite $\mathbb{S}(M_{12}^{11}) \approx 250,000$. We confirmed this by also measuring $\mathbb{P}(M_{12}^{11}, 20,000)$ empirically.



(a) $t = 12500$ **(b)** $t = 20000$

Figure 9: Training runs of M_{12}^{11} varying data seeds after training with seed 11 for t steps, where all runs converge after step t regardless of training seed

We performed this experiment on a larger scale (30 training runs for each t), and found the following result:

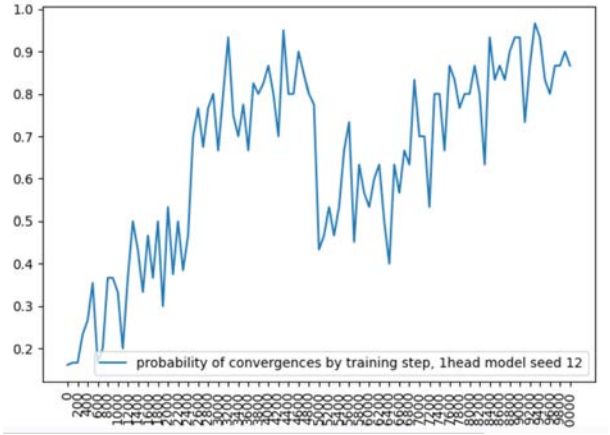


Figure 10: $\mathbb{P}(M_{12}^{11}, t)$, with t on the x-axis and $\mathbb{P}(M_{12}^{11}, t)$ on the y-axis, where we see a continual increase of probability of convergence as training happens

We found a similar curve for $\mathbb{P}(M_{25}^{12}, t)$, where $\mathbb{S}(M_{25}^{12}) \approx 250000$ also:

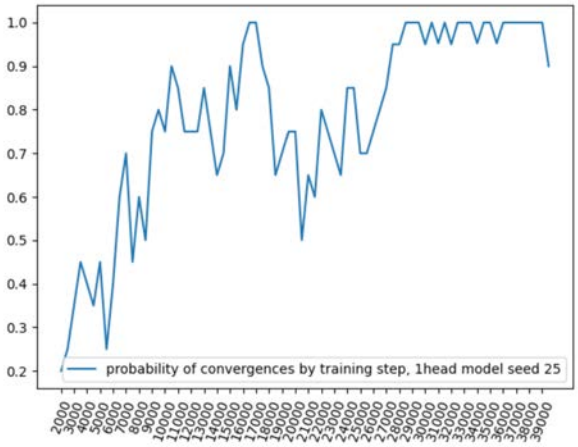


Figure 11: $\mathbb{P}(M_{25}^{12}, t)$, with t on the x-axis and $\mathbb{P}(M_{25}^{12}, t)$ on the y-axis, where we see a continual increase of probability of convergence as training happens

We also observed $\mathbb{P}(M_{12}^{11}, t)$ against $Median(\mathbb{S}(M_{12}^{11}, t))$, the median step at which M_{12}^{11} converges, if at all, after training with seed 11 for t steps and switching to a different seed. We found no correlation between the two quantities.

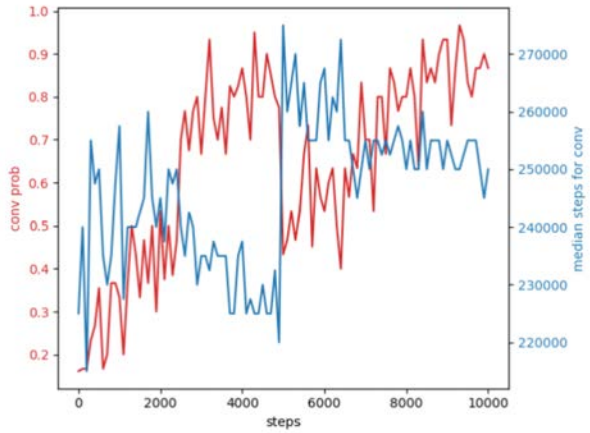


Figure 12: $\mathbb{P}(M_{12}^{11}, t)$ and $Median(\mathbb{S}(M_{12}^{11}, t))$, with t on the x-axis, where the number of steps for model convergence exhibits no correlation to its probability of convergence

We have also found that $200000 < Min(\mathbb{S}(M_{12}^{11}, t)) < Max(\mathbb{S}(M_{12}^{11}, t)) < 290000$.

Finally, we have observed that locally, for $t \in [2400, 2420]$, the variance of $\mathbb{P}(M_{12}^{11}, t)$ is non-trivial.

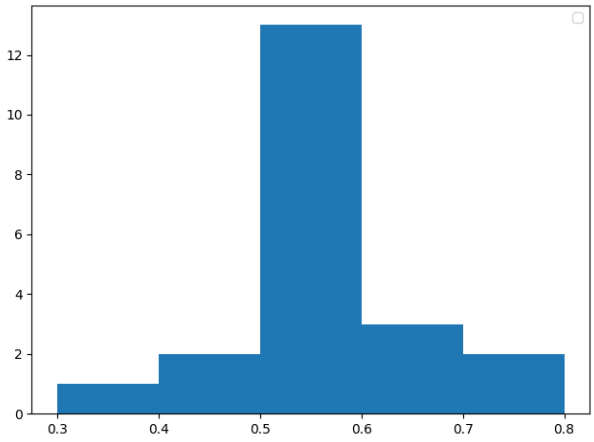


Figure 13: Distribution of $\mathbb{P}(M_{12}^{11}, t)$, for $t \in [2400, 2420]$, where the variance of the probabilities is non-trivial

That is, even if the limiting behavior seems determined, since $\mathbb{P}(M_{12}^{11}, 2400) \approx 0.8$, locally it is still unpredictable.

Observations

We can see that for both of these models,

$$\mathbb{P}(M_i^j, t) \rightarrow 1 \text{ when } t << \mathbb{S}(M_i^j) \tag{.22}$$

that is, the model's "fate" of convergence is determined very early on during training, despite its actual convergent steps being much later. Unfortunately, both attention patterns and the model l_2 norm fail to predict such "fates" of convergence.

5 Conclusion, Future Directions

We have found that even in small and toy transformer models, the training dynamic and its behavior of converging to optimal solutions can be complex to understand and model. These results suggest that there may exist underlying probability distributions that may model convergence behavior.

5.1 Result 1, Future Directions

We suggest that future research can proceed in the direction of introducing more diverse tasks to the simple transformer models and investigating whether such phase change behaviors are still prominent in these tasks, especially for when they can algorithmically achieve accuracy 1. We also suggest, in the light of recent introductions to large language models, investigations on whether such phase change behaviors are prominent on larger models as well, and whether such phase change behaviors still indicate emergent abilities.

5.2 Result 2, Future Directions

We suggest that future research can proceed in the direction of detecting and predicting attention patterns across different transformer models (that are simple enough for such manual interpretations of attention patterns). We also suggest research in determining the landscape of possible attention patterns in which a given task may be solved in generalized 100% accuracy, as well as "forcing" backpropagation into those directions such that these solutions may be achieved earlier during training.

5.3 Result 3, Future Directions

We suggest future research directions to look for different indicators of model performance that are correlated with emergent abilities. We also welcome theoretic interpretations of these indicators on the model's mechanic behavior itself, and further theorizing of metric-mechanic relationships beyond determining whether emergent abilities are present.

5.4 Result 4, Future Directions

We suggest future research directions in characterizing the underlying random processes (if such a process exists) under which the model determines its convergence probability. This may shed light on the scholastic gradient descent algorithm, and help the machine learning community form better theories on why and how models converge to the specific solutions that they do.

Bibliography

1. E. Akyürek, D. Schuurmans, J. Andreas, T. Ma, and D. Zhou. *What learning algorithm is in-context learning? Investigations with linear models.* 2022. arXiv: 2211.15661 [cs.LG].
2. J. L. Ba, J. R. Kiros, and G. E. Hinton. *Layer Normalization.* 2016. arXiv: 1607.06450 [stat.ML].
3. B. Barak, B. L. Edelman, S. Goel, S. Kakade, E. Malach, and C. Zhang. *Hidden Progress in Deep Learning: SGD Learns Parities Near the Computational Limit.* 2023. arXiv: 2207.08799 [cs.LG].
4. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. *Language Models are Few-Shot Learners.* 2020. arXiv: 2005.14165 [cs.CL].
5. Y. Chen, R. Zhong, S. Zha, G. Karypis, and H. He. "Meta-learning via Language Model In-context Tuning". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 719–730. DOI: 10.18653/v1/2022.acl-long.53. URL: https://aclanthology.org/2022.acl-long.53.
6. S. Garg, D. Tsipras, P. Liang, and G. Valiant. *What Can Transformers Learn In-Context? A Case Study of Simple Function Classes.* 2023. arXiv: 2208.01066 [cs.CL].
7. N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt. *Progress measures for grokking via mechanistic interpretability.* 2023. arXiv: 2301.05217 [cs.LG].
8. A. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra. *Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets.* 2022. arXiv: 2201.02177 [cs.LG].
9. R. Schaeffer, B. Miranda, and S. Koyejo. *Are Emergent Abilities of Large Language Models a Mirage?* 2023. arXiv: 2304.15004 [cs.AI].
10. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. *Attention Is All You Need.* 2017. arXiv: 1706.03762 [cs.CL].
11. J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus. *Emergent Abilities of Large Language Models.* 2022. arXiv: 2206.07682 [cs.CL].