Effects of Parameter Norm Growth During Transformer Training: Inductive Bias from Gradient Descent

William Merrill[*†] **Vivek Ramanujan**[*‡] **Yoav Goldberg**[*§] **Roy Schwartz**[¶] **Noah A. Smith**[*‡]

[*] Allen Institute for AI   [†] New York University   [‡] University of Washington
[§] Bar Ilan University   [¶] Hebrew University of Jerusalem

willm@nyu.edu ramanv@cs.washington.edu
{yoavg,noah}@allenai.org roys@cs.huji.ac.il

Abstract

The capacity of neural networks like the widely adopted transformer is known to be very high. Evidence is emerging that they learn successfully due to inductive bias in the training routine, typically a variant of gradient descent (GD). To better understand this bias, we study the tendency for transformer parameters to grow in magnitude (ℓ_2 norm) during training, and its implications for the emergent representations within self attention layers. Empirically, we document norm growth in the training of transformer language models, including T5 during its pretraining. As the parameters grow in magnitude, we prove that the network approximates a discretized network with saturated activation functions. Such "saturated" networks are known to have a reduced capacity compared to the full network family that can be described in terms of formal languages and automata. Our results suggest saturation is a new characterization of an inductive bias implicit in GD of particular interest for NLP. We leverage the emergent discrete structure in a saturated transformer to analyze the role of different attention heads, finding that some focus locally on a small number of positions, while other heads compute global averages, allowing counting. We believe understanding the interplay between these two capabilities may shed further light on the structure of computation within large transformers.

1 Introduction

Transformer-based models (Vaswani et al., 2017) like BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019), and T5 (Raffel et al., 2019) have pushed the state of the art on an impressive array of NLP tasks. Overparameterized transformers are known to be unversal approximators (Yun et al., 2020), suggesting their generalization performance ought to rely on useful biases or constraints imposed by the learning algorithm. Despite various attempts to study these biases in transformers (Rogers et al., 2020; Lovering et al., 2021), it remains an interesting open question what they are, or even how to characterize them in a way relevant to the domain of language.

In this work, we take the perspective that thoroughly understanding the dynamics of gradient descent (GD) might clarify the linguistic biases of transformers, and the types of representations they acquire. We start by making a potentially surprising empirical observation (§3): the parameter ℓ_2 norm grows proportional to $\sqrt{t}$ (where t is the timestep) during the training of T5 (Raffel et al., 2019) and other transformers. We refer to the phenomenon of growing parameter norm during training as *norm growth*. Previous work has analyzed norm growth in simplified classes of feedforward networks (Li and Arora, 2019; Ji and Telgarsky, 2020), but, to our knowledge, it has not been thoroughly demonstrated or studied in the more complicated and practical setting of transformers.

Our main contribution is analyzing the *effect* of norm growth on the representations within the transformer (§4), which control the network's grammatical generalization. With some light assumptions, we prove that any network where the parameter norm diverges during training approaches a *saturated* network (Merrill et al., 2020): a restricted network variant whose discretized representations are understandable in terms of formal languages and automata. Empirically, we find that internal representations of pretrained transformers approximate their saturated counterparts, but for randomly initialized transformers, they do not. This suggests that the norm growth implicit in training guides transformers to approximate saturated networks, justifying studying the latter (Merrill, 2019) as a way to analyze the linguistic biases of NLP architectures and the structure of their representations.

Past work (Merrill, 2019; Bhattamishra et al., 2020) reveals that saturation permits two useful types of attention heads within a transformer: one

that locally targets a small number of positions, and one that attends uniformly over the full sequence, enabling an "average" operation. Empirically, we find that both of these head types emerge in trained transformer language models. These capabilities reveal how the transformer can process various formal languages, and could also suggest how it might represent the structure of natural language. Combined, our theoretical and empirical results shed light on the linguistic inductive biases imbued in the transformer architecture by GD, and could serve as a tool to analyze transformers, visualize them, and improve their performance.

Finally, we discuss potential causes of norm growth in §5. We prove transformers are approximately *homogeneous* (Ji and Telgarsky, 2020), a property that has been extensively studied in deep learning theory. With some simplifying assumptions, we then show how homogeneity might explain the $\sqrt{t}$ growth observed for T5.[1]

2 Background and Related Work

2.1 GD and Deep Learning Theory

A simple case where deep learning theory has studied the generalization properties of GD is matrix factorization (Gunasekar et al., 2017; Arora et al., 2019; Razin and Cohen, 2020). It has been observed that deep matrix factorization leads to low-rank matrix solutions. Razin and Cohen (2020) argued theoretically that this bias of GD cannot be explained as an implicit regularizer minimizing some norm. Rather, they construct cases where all parameter norms diverge during GD.

Similar ideas have emerged in recent works studying feedforward networks. Analyzing biasless ReLU networks with cross-entropy loss, Poggio et al. (2019, 2020) show that the *magnitude* (ℓ_2 norm) of the parameter vector continues to grow during GD, while its *direction* converges. Li and Arora (2019) present a similar argument for *scale-invariant* networks, meaning that scaling the parameters by a constant does not change the output. Studying *homogeneous* networks, Ji and Telgarsky (2020) show that the gradients become *aligned* as $t \to \infty$, meaning that their direction converges to the parameter direction. This means the norm will grow monotonically with t. The perspective developed by these works challenges the once conventional wisdom that the parameters

converge to a finite local minimum during GD training. Rather, it suggests that GD follows a norm-increasing *trajectory* along which network behavior stabilizes. These analyses motivate investigation of this trajectory-driven perspective of training.

From a statistical perspective, work in this vein has considered the implications of these training dynamics for margin maximization (Poggio et al., 2019; Nacson et al., 2019; Lyu and Li, 2019). While these works vary in the networks they consider and their assumptions, they reach similar conclusions: GD follows trajectories diverging in the direction of a max-margin solution. As margin maximization produces a simple decision boundary, this property suggests better generalization than an arbitrary solution with low training loss. This point of view partially explains why growing norm is associated with better generalization performance.

2.2 NLP and Formal Language Theory

Norm growth has another interpretation for NLP models. Past work characterizes the capacity of infinite-norm networks in terms of formal languages and automata theory. Merrill (2019) and Merrill et al. (2020) propose *saturation*, a framework for theoretical analysis of the capacity of NLP architectures. A network is analyzed by assuming it *saturates* its nonlinearities, which means replacing functions like σ and tanh with step functions. This is equivalent to the following definition:

Definition 1 (Saturation; Merrill et al., 2020) Let $f(x; \theta)$ be a neural network with inputs x and weights θ. The *saturated network* $sf(x; \theta)$ is[2]

$$sf(x; \theta) = \lim_{c \to \infty} f(x; c\theta),$$

where the limit exists, and undefined elsewhere.

Saturation reduces continuous neural networks to discrete computational models resembling automata or circuits, making some kinds of formal linguistic analysis easier. For many common architectures, the saturated capacity is known to be significantly weaker than the full capacity of the network with rational-valued weights (Merrill, 2019), which, classically, is Turing-complete for even simple RNNs (Siegelmann and Sontag, 1992).

For example, one can hand-construct an RNN or LSTM encoding a stack in its recurrent memory (Kirov and Frank, 2012). Stacks are useful for processing compositional structure in linguistic data

---

[2]The limit over f is taken pointwise. The range of sf is $\mathbb{R}$.

(Chomsky, 1956), e.g., for semantic parsing. However, a saturated LSTM does not have enough memory to simulate a stack (Merrill, 2019). Rather, saturated LSTMs resemble classical counter machines (Merrill, 2019): automata limited in their ability to model hierarchical structure (Merrill, 2020). Experiments suggest that LSTMs trained on synthetic tasks learn to implement counter memory (Weiss et al., 2018; Suzgun et al., 2019a), and that they fail on tasks requiring stacks and other deeper models of structure (Suzgun et al., 2019b). Similarly, Shibata et al. (2020) found that LSTM language models trained on natural language data acquire saturated representations approximating counters.

Recent work extends saturation analysis to transformers (Merrill, 2019; Merrill et al., 2020). Saturated attention heads reduce to generalized hard attention, where the attention scores can tie. In the case of ties, the head output averages the positions with maximal scores.[3] While their power is not fully understood, saturated transformers can implement a counting mechanism similarly to LSTMs (Merrill et al., 2020). In practice, Bhattamishra et al. (2020) show transformers can learn tasks requiring counting, and that they struggle when more complicated structural representations are required. Ebrahimi et al. (2020) find that attention patterns of certain heads can emulate bounded stacks, but that this ability falls off sharply for longer sequences. Thus, the abilities of trained LSTMs and transformers appear to be predicted by the classes of problems solvable by their saturated counterparts. Merrill et al. (2020) conjecture that the saturated capacity might represent a class of tasks implicitly learnable by GD, but it is unclear *a priori* why this should be the case. This work aims to put this conjecture on more solid theoretical footing: we argue that approximate saturation arises in transformers as a result of norm growth during training.[4]

3 Norm Growth in Transformers

We start with the observation that the parameter ℓ_2 norm grows during training for practical transformer language models. We first consider the parameter norm of 104 historical checkpoints from T5-base (Raffel et al., 2019) pretraining, a 220M

parameter model, which was trained using the AdaFactor optimizer (Shazeer and Stern, 2018). Further details are in §A.

Fig. 1 shows that the T5 norm follows a $\sqrt{t}$ trend, where t is time in training steps. The top right of Fig. 1 breaks down the growth trend by layer. Generally, the norm grows more quickly in later layers than in earlier ones, although always at a rate proportional to $\sqrt{t}$.[5] Next, in the bottom row of Fig. 1, we plot the cosine similarity between each parameter checkpoint θ_{t+1} and its predecessor θ_t. This rapidly approaches 1, suggesting the "direction" of the parameters ($\theta_t/\|\theta_t\|$) converges. The trend in directional convergence looks similar across layers.

We also train smaller transformer language models with 38M parameters on Wikitext-2 (Merity et al., 2016) and the Penn Treebank (PTB; Marcus et al., 1993). We consider two variants of the transformer: pre-norm and post-norm, which vary in the relative order of layer normalization and residual connections (cf. Xiong et al., 2020). Every model exhibits norm growth over training.[6]

Combined, these results provide evidence that the parameter norm of transformers tends to grow over the course of training. In the remainder of this paper, we will discuss the implications of this phenomenon for the linguistic biases of transformers, and then discuss potential causes of the trend rooted in the optimization dynamics.

4 Effect of Norm Growth

§3 empirically documented that the parameter norm grows proportional to $\sqrt{t}$ during T5 pretraining. Now, we move to the main contribution of our paper: the implications of norm growth for understanding transformers' linguistic inductive biases. In particular, Prop. 1 says uniform norm growth across the network guides GD towards saturated networks. Thus, saturation is not just a useful approximation for analyzing networks, but a state induced by training with enough time.

Proposition 1 (Informal) *Let $\theta_t \in \mathbb{R}^n$ be parameters at step t for $f(x; \theta_t)$. If every scalar parameter θ_t^i diverges at the same rate up to a constant, then f converges pointwise to a saturated network.*

---

[3] Hahn (2020) identified weaknesses of *strictly* hard attention, which is weaker than saturated attention.

[4] This relates to Correia et al. (2019), who modify the transformer to facilitate approximately sparse attention. In contrast, we will show that approximate sparsity (i.e, saturation) arises implicitly in *standard* transformers.

[5] We encourage future works that pretrain new transformer language models to track metrics around norm growth.

[6] The post-norm transformer achieves 115.79 perplexity on Wikitext-2 and 96.24 on PTB. On the other hand, the pre-norm transformer reaches 66.35 on Wikitext-2 and 26.16 on PTB, slightly outperforming Wang et al. (2019). This is consistent with previous findings (Xiong et al., 2020) showing advantages of pre-norm over post-norm.
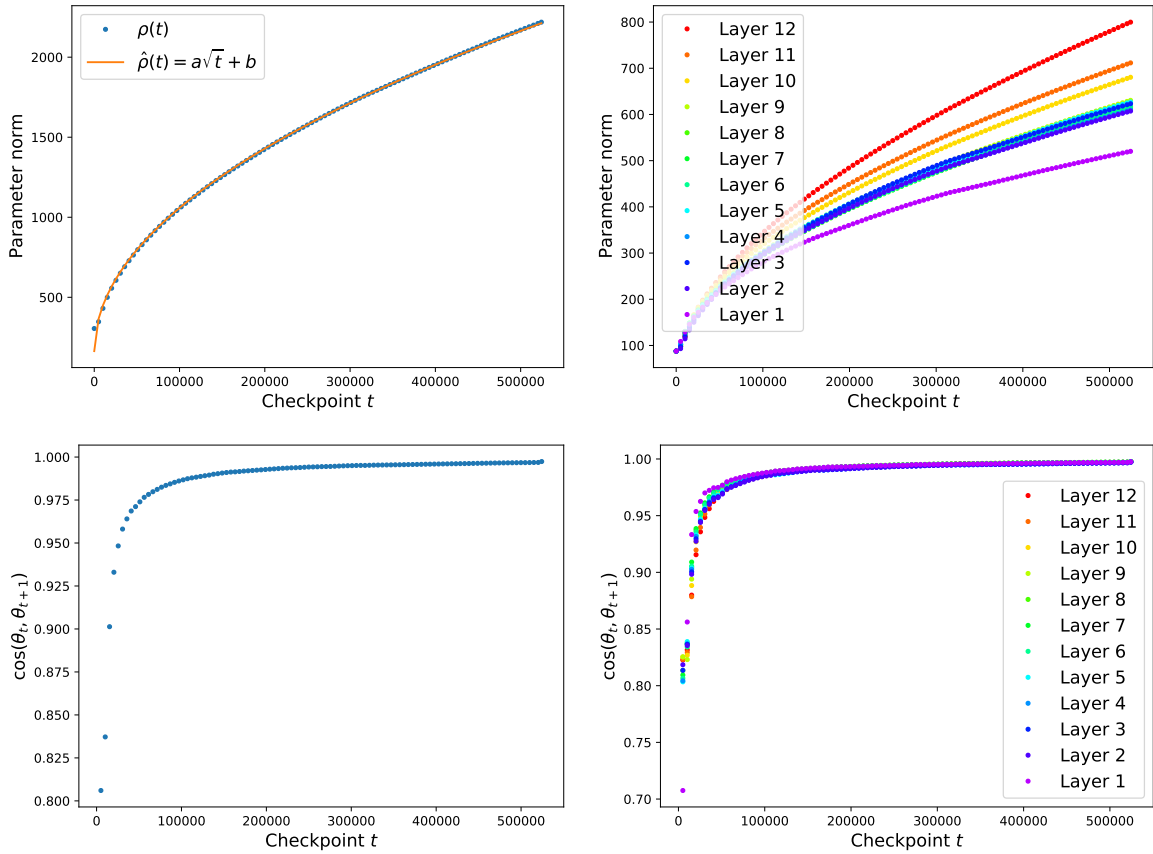
Figure 1: **Top**: Norm growth during T5 pretraining, with a coefficient $r^2 = 1.00$. The right is broken down by layer. **Bottom**: cosine similarity between subsequent parameter checkpoints.

The proof is in §B. Prop. 1 assumes not just norm growth, but uniform norm growth, meaning no parameter can asymptotically dominate any other. Notably, uniform growth implies directional convergence. Accepting uniform growth for a given training regimen, we expect transformers to converge to saturated networks with infinite training. Based on §3, the T5 norm appears to grow $\propto \sqrt{t}$ uniformly across the network, suggesting the uniform growth condition is reasonable. As we will discuss later in §5, we expect the growth trend to depend heavily on the learning rate schedule.

4.1 Saturated Transformers

Having established that norm growth should lead to saturation, we now empirically measure the saturation levels in T5 and other transformer models.

Large transformers are highly saturated. Since $\|\theta_t\|$ empirically grows during training, we expect high cosine similarity between the representations in trained networks and saturated representations. We estimate this as the cosine similarity between $f(x; \theta)$ and $f(x; c\theta)$ for some

large c (in practice, 1,000). We consider the "base" versions of pretrained BERT, RoBERTa, T5, and XLNet (pretrained on masked language modeling), and compute the mean saturation over 100 input sentences from the Brown corpus (Francis and Kučera, 1989). To match standard practice, each sentence is truncated at 512 word pieces. Fig. 2 plots the similarity for each layer of each model. We compare the pretrained transformers against a randomly initialized baseline. For every model type, the similarity is higher for the pretrained network than the randomly initialized network, which, except for T5, is ~0. For T5 and XLNet, the similarity in the final layer is ≥0.9, whereas, for RoBERTa, the final similarity is 0.65 (although 0.94 in the penultimate layer). For T5 and XLNet, similarity is higher in later layers, which is potentially surprising, as one might expect error to compound with more layers. This may relate to the fact that the norm grows faster for later layers in T5. One question is why the similarity for BERT is lower than these models. As RoBERTa is architecturally similar to BERT besides longer
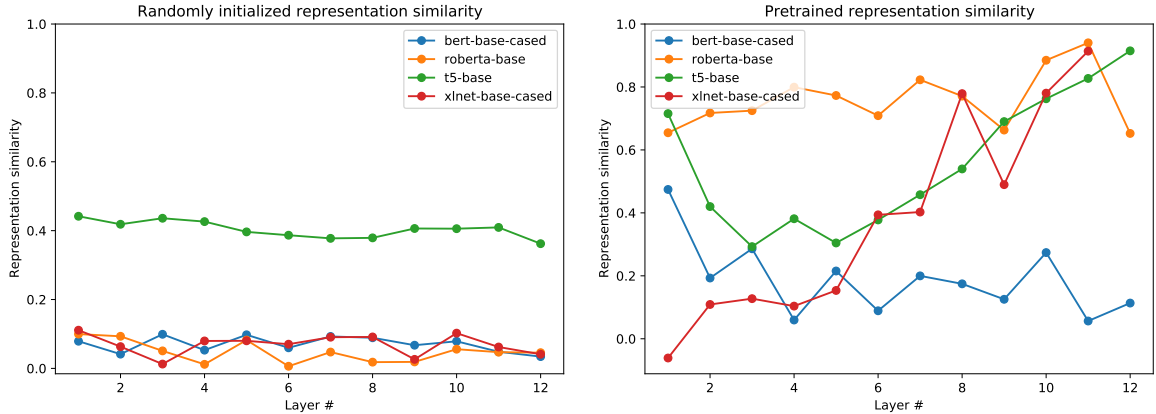
Figure 2: Cosine similarities of the unsaturated and saturated ($c = 1{,}000$) transformer representations, by layer. We compare randomly initialized transformers (**left**) to pretrained ones (**right**).

training, we hypothesize that RoBERTa's higher similarity is due to longer pretraining.

Small transformers reach full saturation. Each of the transformers trained on Wikitext-2 and PTB reached a saturation level of 1.00. It is unclear why these models saturate more fully than the pretrained ones, although it might be because they are smaller.[7] For our LMs, the feedforward width (512) is less than for T5-base, while the encoder depth and width are the same. Other possible explanations include differences in the initialization scheme, optimizer, and training objective (masked vs. next-word modeling). See §A for full hyperparameters.

4.2 Power of Saturated Attention

We have shown that transformer training increases the parameter norm (§3), creating a bias towards saturation (§4.1). Now, we discuss the computational capabilities of saturated transformers, and empirically investigate how they manifest in pretrained transformers. What computation can saturated transformers perform? We review theoretical background about saturated attention, largely developed by Merrill (2019). Let H (sequence length n by model dimension d) be the input representation to a self attention layer. We assume a standard self attention mechanism with key, query, and value matrices K, Q, V.[8] Saturated attention resembles standard attention where softmax is constrained to a generalization of "argmax" (Merrill, 2019):

$$\mathrm{s\,attn}(H; Q, K, V) = \arg\max(HQK^\top H^\top)HV.$$

[7]Qualitatively, we observed that *-small transformers tended to be more saturated than the *-base models.

[8]To simplify presentation, we omit bias terms.

We define this vectorized $\arg\max(A)$ as

$$\mathcal{M}(A_i) = \{j \mid a_{ij} = \max_k a_{ik}\}$$

$$\arg\max(A_i)_j = \begin{cases} 1/|\mathcal{M}(A_i)| & \text{if } j \in \mathcal{M}(A_i) \\ 0 & \text{otherwise.} \end{cases}$$

Crucially, in the case of ties, $\arg\max(A)$ returns a uniform distribution over all tied positions. Saturated attention can retrieve the "maximum" value in a sequence according to some similarity matrix. It is also capable of restricted counting (Merrill et al., 2020). Formalizing these observations, we identify two useful computational operations that are reducible to saturated self attention: *argmax* and *mean*. Let h_i represent the input representation at each time step $1 \leq i \leq n$.

1. **Argmax:** Set $V = \mathrm{Id}$. Then the self attention mechanism computes a function recovering the element of H that maximally resembles h_i according to a quadratic form $M = KQ^\top$. If there is a tie for similarity, a uniform average of the maximal entries in H is returned.

$$\mathrm{argmax}(H; M) = \arg\max_j h_i M h_j^\top.$$

2. **Mean:** Parameterize the head to attend uniformly everywhere. Then the head computes a function taking a uniform average of values:

$$\mathrm{mean}(H; V) = \frac{1}{n} \sum_{j=1}^n V h_j. \qquad (1)$$

These constructions demonstrate some useful computational abilities of saturated transformers. Due to the summation in (1), the mean operation (or near variants of it) can be used to implement
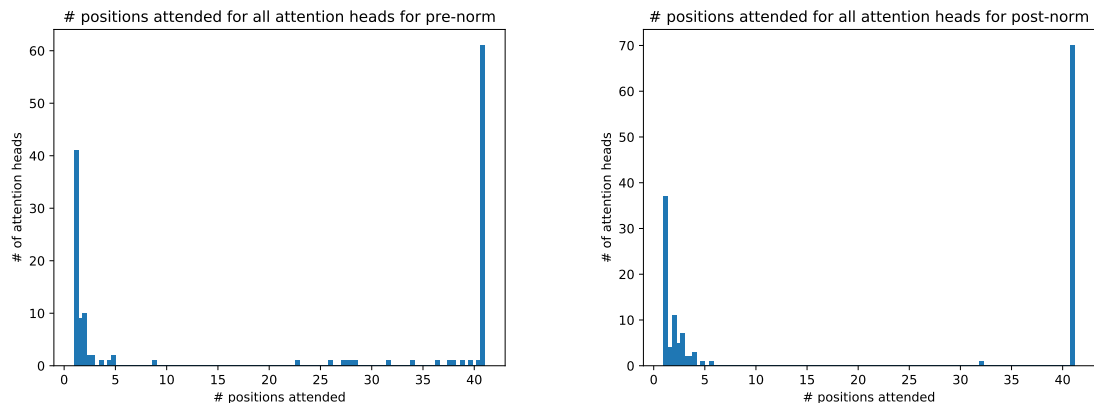
1770

Figure 3: Distribution of the number of positions attended to for all heads in the PTB language models. The **left** plot is pre-norm, and the **right** is post-norm. Values are averaged over 200 sentences from the development set.

counting, which allows recognizing languages like $a^n b^n c^n$ (Merrill et al., 2020). Empirically, Bhattamishra et al. (2020) find trained networks can learn to recognize counter languages that rely on computing means, failing on more complicated languages like Dyck-2. Our findings partially justify why transformers can learn these languages: they lie within the capacity of *saturated* transformers.

4.3 Learned Attention Patterns

Recall that the small language models trained in §4.1 reach 1.00 saturation. It follows that we can convert them to saturated transformers (by multiplying θ by a large constant c) without significantly shifting the representations in cosine space. We will evaluate if the saturated attention heads manifest the argmax and mean constructions from §4.2.

As discussed in §4.2, saturated attention can parameterize both argmax and mean heads. An argmax head should attend to a small number of positions. A mean head, on the other hand, attends uniformly over the full sequence. Are both patterns acquired in practice by our models? We plot the distribution of the number of positions attended to by each head in the saturated PTB models in Fig. 3. The distribution is bimodal, with one mode at 1, and the other around 41, representing the mean sequence length of a 83-length encoder with positional masking to prevent lookahead. The empirical mode around 1 corresponds to heads that are argmax-like. The mode around 41, on the other hand, corresponds to mean-like heads, since it implies uniform attention over the masked sequence. Thus, our analysis suggests that analogs of both types of attention heads theorized in §4.2 are ac-

quired in transformers in practice. In the pre-norm transformer, which performs substantially better, there are also a small number of heads lying between the two modes. We defer the investigation of the function of these heads to future work.

5 Explanation for Norm Growth

We have documented norm growth in T5 and other transformers (§3) and showed how it induces partial saturation in their representations (§4). This section points towards an understanding of *why* the parameter norm grows over the course of training, grounded in results about norm growth from deep learning theory. We do not analyze specific optimizers directly; instead, we analyze norm growth within simplified models of training dynamics taken from the literature. We then evaluate how these candidate dynamics models fit T5's training.

5.1 Setup

Let $\delta_t \in \mathbb{R}^n$ denote the optimizer step at time t, i.e., $\delta_t = \theta_{t+1} - \theta_t$. We write η_t for the learning rate at t.[9] Let $\nabla_{\theta_t} L$ denote the gradient of the loss with respect to θ_t. By GD, we refer to the update $\delta_t = -\eta_t \nabla_{\theta_t} L$.[10] In contrast, we will use the term *gradient flow* to refer to its continuous relaxation, specified by an analogous differential equation:

$$\frac{\mathrm{d}\theta_t}{\mathrm{d}t} = -\eta_t \nabla_{\theta_t} L.$$

---

[9]Without loss of generality, the arguments presented here can be seen as applying to an individual parameter in the network, or the vector of all concatenated network parameters.

[10]Note that, in practice, T5 was trained with AdaFactor, whereas the setup in this section assumes simpler optimizers.
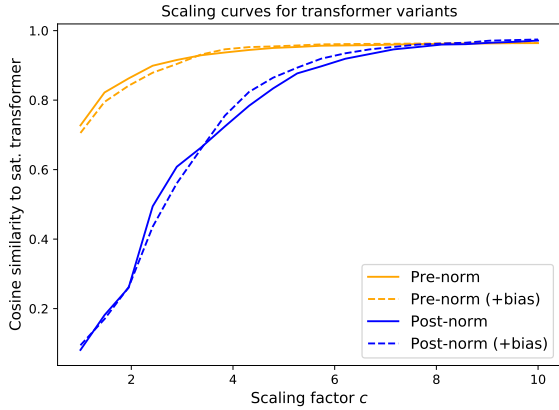
Figure 4: Approximate cosine similarity of $f(x; c\theta)$ to $\mathrm{s}f(x; \theta)$ for randomly initialized transformers f. $\mathrm{s}f(x; \theta)$ is approximated as in Fig. 2.

5.2 Homogeneity

We will rely on properties of homogeneous networks, a class of architectures well-studied in deep learning theory (Ji and Telgarsky, 2020).

Definition 2 (Homogeneity) A function $f(x; \theta)$ is *k-homogeneous* in θ iff, for all $c \geq 0$, $f(x; c\theta) = c^k f(x; \theta)$. We further say that f is homogeneous iff there exists some k such that f is k-homogeneous.

Many common components of modern neural networks are homogeneous (Li and Arora, 2019). Furthermore, as various computations within a neural network preserve homogeneity (§C), some full networks are also homogeneous. An example of a fully homogeneous neural network is a feedforward ReLU network without bias terms.

Why is homogeneity relevant for transformers? Transformers are not homogeneous, but they are *almost* homogeneous. We formalize this as:

Definition 3 (Approx. homogeneity) A scalar[11] function $f(x; \theta)$ is approximately k-homogeneous in θ iff there exist d, ρ s.t., for $c \geq 1$ and $\|\theta\| \geq \rho$,

$$\left| f(x; c\theta) - c^k f(x; \theta) \right| \leq \exp(-d\|\theta\|).$$

In other words, as $\|\theta\|$ grows, f approximates a homogeneous function with exponentially vanishing error. In §D, we prove transformer encoders without biases are approximately 1-homogeneous. In Fig. 4, we compare the cosine similarity of transformers with and without biases to their saturated variants, as a function of a constant c scaling their weights. An approximately homogeneous function

rapidly approach 1.0 as c increases. We find similar curves for transformers with and without biases, suggesting biasless transformers are similarly homogeneous to transformers with biases.[12]

Since multiplying two homogeneous functions adds their homogeneity, a transformer encoder followed by a linear classifier is approximately 2-homogeneous. A key property of homogeneous functions is Euler's Homogeneity Theorem: the derivative of a k-homogeneous function is $(k-1)$-homogeneous. Thus, we will assume the gradients of the linear classifier output are roughly 1-homogeneous, which under simple GD implies:

Assumption 1 *Let θ_t include all encoder and classifier parameters. Let $\asymp$ mean "approximately proportional to". For large enough t during transformer training, $\|\delta_t\| \asymp \eta_t \|\theta_t\|$.*

5.3 Aligned Dynamics

We now consider the first candidate dynamics model: **aligned dynamics** (Ji and Telgarsky, 2020). Analyzing homogeneous networks with an exponential binary classification loss and gradient flow, Ji and Telgarsky (2020) show that the parameters converge in direction, and that the gradients become *aligned*, meaning that $\theta_t^\top \cdot \delta_t \to \|\theta_t\| \|\delta_t\|$. While it is unclear whether transformers will follow aligned dynamics, we entertain this as one hypothesis. Under Ass. 1, alignment implies

$$\|\theta_t\| \approx \sum_{i=0}^{t} \|\delta_i\| \asymp \int \eta_t \|\theta_t\| \mathrm{d}t.$$

With the $\eta_t = 1/\sqrt{t}$ schedule used by T5 (Raffel et al., 2019), $\|\theta_t\| \asymp \exp(\sqrt{t})$ (see §E.1). This is asymptotically faster than the observed $\sqrt{t}$ growth, suggesting an alternate dynamics might be at play.

5.4 Misaligned Dynamics

Our second candidate model of training is **misaligned dynamics**, which follows largely from Li and Arora (2019). This can be derived by assuming the gradients are misaligned (i.e., $\theta_t^\top \cdot \delta_t = 0$), which hold for scale-invariant networks (Li and Arora, 2019) and in expectation for random normal gradients. Misalignment implies (derived in §E.2):

$$\|\theta_t\|^2 \asymp \sum_{i=0}^{t} \|\delta_i\|^2. \tag{2}$$

---

[11] A vector function is approximately k-homogeneous if this holds for all its elements.

[12] Lyu and Li (2019) find similar results for feedforward ReLU networks. It is an interesting puzzle why networks with biases appear similarly homogeneous to those without biases.
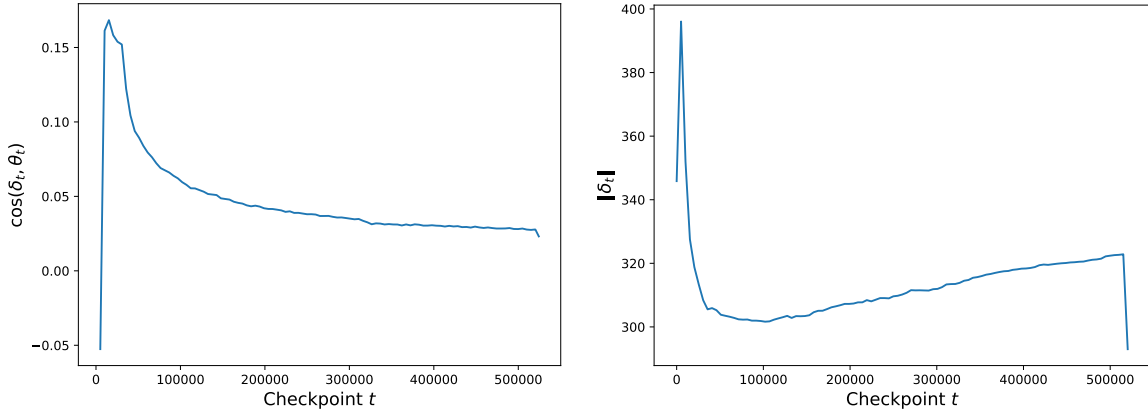
Figure 5: Alignment (cosine similarity of δ_t and θ_t) and step size ($\|\delta_t\|$) over training.

We show in §E.2 that, with the T5 learning rate ($\eta_t = 1/\sqrt{t}$), (2) reduces to $\|\theta_t\| \propto \sqrt{t}$, as observed empirically for T5. We now further test whether misaligned dynamics are a good fit for T5.

5.5 Evaluation

We measure the gradient alignment over the course of training T5. Our alignment metric is the cosine similarity of δ_t to θ_t. As shown on the left of Fig. 5, the alignment initially rapidly increases to ~ 0.15, and then decays to near 0. This supports the hypothesis that the T5 dynamics are misaligned, since the similarity is never high, and may be approaching 0.

On the right of Fig. 5, we plot step size over training in order to evaluate the validity of Ass. 1. At the beginning of training, a chaotic step size seems reasonable, as it is hard to predict the dynamics before approximate homogeneity takes hold. For large t, Ass. 1 combined with the T5 learning rate schedule predicts step size should be roughly constant.[13] This is not exactly what we find: for large t, $\|\delta_t\|$ grows gradually with t. However, the absolute change in step size is small: < 20 across 220M parameters. Thus, we believe Ass. 1 is not unreasonable, though it would be interesting to understand what properties of the optimizer can explain the slight growth in step size.[14]

5.6 Weight Decay

One feature of practical training schemes not considered in this section is weight decay. When applied to standard GD, weight decay can be written $\delta_t = -\eta_t \nabla_{\theta_t} L - \lambda \theta_t$. Intuitively, it might hinder norm growth if λ is large.[15] In §F, we report preliminary experiments testing the effect of weight decay on norm growth. Indeed, if λ is set too large, weight decay can prevent norm growth, but within the standard range of values for λ, we find norm growth even in the face of weight decay. However, it is possible these results may change if the optimizer or other hyperparameters are varied.

6 Conclusion

We empirically found that $\|\theta_t\|$ grows $\propto \sqrt{t}$ during T5 pretraining—a fact that may be caused by the approximate homogeneity of the transformer architecture. We proved that norm growth induces saturation, and then showed empirically that T5 and other large transformers become approximately saturated through their pretraining. Examining highly saturated transformer language models, we found the attention heads largely split between two distinct behaviors that can be roughly interpreted as argmax and mean operations. While we lack a precise formal characterization of "semi-saturated" transformers, we conjecture their capacity resembles that of the saturated models. Thus, we believe further analyzing the capabilities of saturated attention may clarify the linguistic biases that emerge in transformers through training, and the mechanisms they use to represent linguistic structure.

---

[13]Since $\|\delta_t\| \propto \eta_t \|\theta_t\| = \sqrt{t}/\sqrt{t} = 1$.

[14]We believe the sharp drop in $\|\delta_t\|$ at the final step is an artifact of the original recording of these checkpoints.

[15]Common wisdom says that weight decay improves generalization by keeping $\|\theta_t\|$ small; however, recent work challenges the assumption that a bias towards small norm is beneficial (Goldblum et al., 2020), suggesting the benefit of weight decay may arise from more subtle effects on the GD trajectory.

References

Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. 2019. Implicit regularization in deep matrix factorization.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020. On the Ability and Limitations of Transformers to Recognize Formal Languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, Online. Association for Computational Linguistics.

N. Chomsky. 1956. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2174–2184, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Javid Ebrahimi, Dhruv Gelda, and Wei Zhang. 2020. How can self-attention networks recognize Dyck-n languages? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4301–4306, Online. Association for Computational Linguistics.

Winthrop Nelson Francis and Henry Kučera. 1989. *Manual of information to accompany a standard corpus of present-day edited American English, for use with digital computers*. Brown University, Department of Linguistics.

Micah Goldblum, Jonas Geiping, Avi Schwarzschild, Michael Moeller, and Tom Goldstein. 2020. Truth or backpropaganda? an empirical investigation of deep learning theory. In *International Conference on Learning Representations*.

Suriya Gunasekar, Blake Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nathan Srebro. 2017. Implicit regularization in matrix factorization.

Michael Hahn. 2020. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171.

Ziwei Ji and Matus Telgarsky. 2020. Directional convergence and alignment in deep learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 17176–17186. Curran Associates, Inc.

Christo Kirov and Robert Frank. 2012. Processing of nested and cross-serial dependencies: an automaton perspective on SRN behaviour. *Connection Science*, 24(1):1–24.

Zhiyuan Li and Sanjeev Arora. 2019. An exponential learning rate schedule for deep learning. In *Proc. of ICLR*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Charles Lovering, Rohan Jha, Tal Linzen, and Ellie Pavlick. 2021. Predicting inductive biases of pretrained models. In *International Conference on Learning Representations*.

Kaifeng Lyu and Jian Li. 2019. Gradient descent maximizes the margin of homogeneous neural networks.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

William Merrill. 2019. Sequential neural networks as automata. *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*.

William Merrill. 2020. On the linguistic capacity of real-time counter automata.

William. Merrill, Gail Garfinkel Weiss, Yoav Goldberg, Roy Schwartz, Noah A. Smith, and Eran Yahav. 2020. A formal hierarchy of RNN architectures. In *Proc. of ACL*.

Mor Shpigel Nacson, Suriya Gunasekar, Jason D. Lee, Nathan Srebro, and Daniel Soudry. 2019. Lexicographic and depth-sensitive margins in homogeneous and non-homogeneous deep models.

Tomaso Poggio, Andrzej Banburski, and Qianli Liao. 2019. Theoretical issues in deep networks: Approximation, optimization and generalization.

Tomaso Poggio, Qianli Liao, and Andrzej Banburski. 2020. Complexity control by gradient descent in deep networks. *Nature communications*, 11(1):1–5.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer.

Noam Razin and Nadav Cohen. 2020. Implicit regularization in deep learning may not be explainable by norms.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *CoRR*, abs/1804.04235.

Chihiro Shibata, Kei Uchiumi, and Daichi Mochihashi. 2020. How LSTM encodes syntax: Exploring context vectors and semi-quantization on natural text.

Hava T. Siegelmann and Eduardo D. Sontag. 1992. On the computational power of neural nets. In *Proc. of COLT*, pages 440–449.

Mirac Suzgun, Yonatan Belinkov, Stuart Shieber, and Sebastian Gehrmann. 2019a. LSTM networks can perform dynamic counting. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 44–54.

Mirac Suzgun, Sebastian Gehrmann, Yonatan Belinkov, and Stuart M. Shieber. 2019b. Memory-augmented recurrent neural networks can learn generalized Dyck languages.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Chenguang Wang, Mu Li, and Alexander J. Smola. 2019. Language models with transformers. *CoRR*, abs/1904.09408.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision RNNs for language recognition.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. 2020. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*.

A Experimental Details

We provide experimental details for the small language models that we trained. The models were trained for 5 epochs, and the best performing model was selected based on development loss. Reported metrics were then measured on the held-out test set. We used our own implementation of the standard pre- and post-norm transformer architectures. We did not do any hyperparameter search, instead choosing the following hyperparameters:

- Batch size of 16
- Model dimension of 768
- Feedforward hidden dimension of 512
- 12 heads per layer
- 12 layers
- AdamW optimizer with default PyTorch hyperparameters
- 0 probability of dropout
- Default PyTorch initialization

Tokenization For Wikitext-2, 3 tokens in the whole test dataset were unattested in the training set (due to capitalization). To make our model compatible with unseen tokens, we replaced these tokens with <unk>, the same class that appeared for low frequency words at training time, when evaluating the final text perplexity. Due to the small number of tokens that were affected, the impact of this change should be negligible.

Compute We estimate the experiments in this paper took several hundred GPU hours on NVIDIA A100 GPUs over the course of almost two years of on-and-off research time.

T5 We used the historical checkpoints of `bsl-0`, one of five T5-base models that was trained for the original paper (Raffel et al., 2019).

Measuring Norms As a systematic choice, all measurements of parameter norm include only *encoder* parameters that are not scalars. We advise other researchers to follow the practice of excluding embedding parameters, as embedding parameters that are infrequently updated may obscure general trends in the network parameters.

Component	k Input	k Output
Linear	k	$k+1$
Bias	1	1
Affine	0	1
LayerNorm	k	0
LayerNorm + Affine	k	1
ReLU	k	k
Sum	(k,k)	k
Product	(k_1, k_2)	$k_1 + k_2$

Table 1: Effects of network components on homogeneity shown by Li and Arora (2019). We write the "k Output" homogeneity as a function of the "k Input" homogeneity. These facts can be applied recursively to compute the homogeneity of a network. We will show that the same facts hold for *approximate* homogeneity.

B Norm Growth and Saturation

Proposition 2 (Formal version of Prop. 1) *Let $\theta_t \in \mathbb{R}^n$ be the parameter vector at train step t for a network $f(x; \theta_t)$. Assume that, as $t \to \infty$, there exists a scalar sequence $c(t) \to \infty$ and fixed vector $\theta' \in (\mathbb{R} \setminus \{0\})^n$ such that, for all t, $\theta_t \to \theta' \cdot c(t)$. Then f converges pointwise to a saturated network in function space.*

Proof.

$$\lim_{t \to \infty} f(x; \theta_t) = \lim_{t \to \infty} f(x; \theta' \cdot c(t)).$$

Now, since $c(t) \to \infty$ and $\theta' \cdot c(t)$ contains no indeterminate elements, we can simplify this to

$$\lim_{c \to \infty} f(x; c\theta') = \mathrm{s}f(x; \theta').$$

$\square$

C Approximate Homogeneity

In this section, we will further develop the notion of approximate homogeneity. We will prove that is consistent. In other words, every function can have at most one degree k of approximate homogeneity. Next, we will show that the useful closure properties applying to full homogeneity also apply to partial homogeneity.

If $f(\theta)$ is approximately k-homogeneous (cf. Def. 3), then $f(c\theta) = c^k f(\theta) + \epsilon$ for some error vector ϵ where, for each i, $|\epsilon_i| \leq \exp(-d\|\theta\|)$), for all c and large enough $\|\theta\|$. We use this ϵ notation throughout this section.

C.1 Consistency

We first prove that approximate homogeneity is consistent: in other words, if a function is both approximately k_1 and k_2-homogeneous, then $k_1 = k_2$. This is an important property for establishing approximate homogeneity as a meaningful notion.

Lemma 1 *Let $k_1, k_2 \in \mathbb{N}$. Assume that f is both approximately k_1 and k_2-homogeneous. Then $k_1 = k_2$.*

Proof. If f is both approximately k_1 and k_2-homogeneous, then we have vanishing terms ϵ_1 and ϵ_2 such that, for all c,

$$f(c\theta) = c^{k_1} f(\theta) + \epsilon_1$$
$$f(c\theta) = c^{k_2} f(\theta) + \epsilon_2.$$

Subtracting both sides yields

$$0 = (c^{k_1} - c^{k_2}) f(\theta) + \epsilon_1 - \epsilon_2$$
$$\therefore \left| c^{k_1} - c^{k_2} \right| = \frac{|\epsilon_1 - \epsilon_2|}{f(\theta)}.$$

The right-hand side vanishes exponentially in $\|\theta\|$ for all c, whereas the left-hand side grows with c unless $k_1 = k_2$. Thus, to satisfy this equation for all c, it must be the case that $k_1 = k_2$. $\square$

C.2 Closure Properties

We now prove that effects of various functions on homogeneity explored by Li and Arora (2019) also translate to approximate homogeneity.

Lemma 2 ReLU *preserves approximate k-homogeneity, i.e., let $f : \mathbb{R}^n \to \mathbb{R}$ be approximately k-homogeneous. Then $\mathrm{ReLU} \circ f$ is approximately k-homogeneous.*

Proof.

$$\mathrm{ReLU}\left(f(c\theta)\right) = \mathrm{ReLU}\left(c^k f(\theta) + \epsilon\right)$$
$$\leq \mathrm{ReLU}\left(c^k f(\theta)\right) + |\epsilon|.$$

Therefore,

$$\left| \mathrm{ReLU}\left(f(c\theta)\right) - \mathrm{ReLU}\left(c^k f(\theta)\right) \right| \leq |\epsilon|.$$

Set $\epsilon' = |\epsilon|$, showing $\mathrm{ReLU}\left(f(\theta)\right)$ is approximately k-homogeneous. $\square$

Lemma 3 *Let f, g be vector-valued functions of θ. If f and g are approximately k-homogeneous, then $f + g$ is approximately k-homogeneous.*

Proof.

$$f(c\theta) + g(c\theta) = c^k f(\theta) + \epsilon_f + c^k g(\theta) + \epsilon_g$$
$$= c^k f(\theta) + c^k g(\theta) + \epsilon',$$

where $\epsilon' = \epsilon_f + \epsilon_g$. Thus,

$$\left| f(c\theta) + g(c\theta) - c^k\big(f(\theta) + g(\theta)\big) \right| \le \epsilon'.$$

$\square$

Lemma 4 *Let f, g be vector-valued functions of θ. If f and g are approximately k_f and k_g-homogeneous, then $f \cdot g$ is approximately $(k_f + k_g)$-homogeneous.*

Proof.

$$f(c\theta) \cdot g(c\theta) = \big(c^k f(\theta) + \epsilon_f\big) \cdot \big(c^k g(\theta) + \epsilon_g\big)$$
$$= c^{k_f+k_g} f(\theta)g(\theta) + c^{k_f} f(\theta)\epsilon_g$$
$$+ c^{k_g} g(\theta)\epsilon_f + \epsilon_f \epsilon_g.$$

We now rewrite the term $c^{k_f} f(\theta)\epsilon_g$ as

$$\frac{\theta g(x;\hat\theta)}{\exp(-d\|\theta\|)} \le \exp\big(-d'\|\theta\|\big).$$

Now, set $\epsilon' = \min(\exp(-d\|\theta\|), \epsilon_f \epsilon_g)$.

$$\left| f(c\theta)g(c\theta) - c^{k_f+k_g} f(\theta)g(\theta) \right| \le \epsilon'.$$

$\square$

The analogous results for linear transformation, bias, and affine transformation directly follow from the results for sum and product in Lem. 3 and Lem. 4.

Finally, we show that layer norm converts a homogeneous function to approximately scale-invariant function. In order to be numerically stable, practical implementations of layer norm utilize a small tolerance term so that the denominator is never zero. We omit this practical detail from our analysis, instead defining the layer norm $\mathrm{LN}(x)$ for $x \in \mathbb{R}^n$ according to

$$\mu(x) = \frac{1}{n}\sum_{i=1}^{n} x_i$$
$$\mathrm{LN}(x)_i = \frac{x_i - \mu(x)}{\|x - \mu(x)\|}.$$

Lemma 5 *Let f be approximately k-homogeneous for some k. Then, $\mathrm{LN}(f)$ is approximately 0-homogeneous.*

Proof. Since addition preserves approximate k-homogeneity, mean (and difference to mean), preserve approximate k-homogeneity. Letting $C = c^k$, we can write

$$f(c\theta) - \mu(f(c\theta)) = C\big(f(\theta) - \mu(f(\theta))\big) + \epsilon.$$

We now apply this to the definition of layer norm to get

$$\mathrm{LN}(f(c\theta))_i = \frac{f(c\theta)_i - \mu(f(c\theta))}{\|f(c\theta) - \mu(f(c\theta))\|}$$
$$= \frac{C\big(f(\theta)_i - \mu(f(\theta))\big) + \epsilon_i}{C\|f(\theta) - \mu(f(\theta))\| + \epsilon}.$$

We show that the difference between this and the unscaled layer norm goes to zero. To simplify notation, we now write $f = f(\theta)$, $\mu = \mu(f(\theta))$, and $\epsilon = \epsilon$ in the left-hand side below:

$$|\mathrm{LN}(f(c\theta))_i - \mathrm{LN}(f(\theta)_i)|$$
$$= \left| \frac{C(f_i - \mu) + \epsilon_i}{C\|f - \mu\| + \epsilon} - \frac{f_i - \mu}{\|f - \mu\|} \right|$$
$$= \left| \frac{\epsilon_i\|f - \mu\| - \epsilon(f_i - \mu)}{C\|f - \mu\|^2 + \epsilon\|f - \mu\|} \right|$$
$$= \left| \frac{\epsilon_i - \epsilon v}{C\|f - \mu\| + \epsilon} \right|$$
$$\le \left| \frac{\epsilon_i - \epsilon v}{\epsilon} \right|.$$

for some $v \in \mathbb{R}^n$ which does not grow with $\|\theta\|$. Thus, setting ϵ' to this final quantity satisfies the definition of approximate 0-homogeneity, i.e. approximate scale invariance. $\square$

C.3 Saturating Activation Functions

We show that the exponentially saturation activation functions σ, $\mathrm{softmax}$, and $\tanh$ are approximately scale-invariant in x, i.e. scaling x has an exponentially diminishing effect on the output. We start by analyzing the simpler sigmoid, and then show that the same result holds for softmax. For completeness, we then present a proof for $\tanh$. We use Θ (not θ) in the standard sense of asymptotic notation.

Lemma 6 *The scaling error for σ vanishes exponentially in the preactivation magnitude, i.e. for all $c \ge 1$,*

$$|\sigma(cx) - \sigma(x)| \le \Theta(\exp(-|x|)).$$

Proof. Assume without loss of generality that $x \neq 0$, as if this is the case, the error is 0. When $x > 0$, we have

$$
\begin{aligned}
|\sigma(cx) - \sigma(x)| &= \sigma(cx) - \sigma(x) \\
&\leq 1 - \sigma(|x|) \\
&= \frac{1}{\exp(|x|) + 1} \\
&= \Theta(\exp(-|x|)).
\end{aligned}
$$

When $x < 0$, we have

$$
\begin{aligned}
|\sigma(cx) - \sigma(x)| &= \sigma(x) - \sigma(cx) \\
&\leq 1 - \sigma(|x|) + 0 \\
&= \Theta(\exp(-|x|)).
\end{aligned}
$$

$\square$

Lemma 7 *The elementwise scaling error for* softmax *vanishes exponentially in the preactivation norm, i.e. for all $c \geq 1$, $x \in \mathbb{R}^n$ s.t. $1 \leq i \leq n$,*

$$|\text{softmax}(cx)_i - \text{softmax}(x)_i| \leq \exp(-\Theta(\|x\|)).$$

Proof. The proof closely follows that of Lem. 6, but is more involved. We consider two cases: $x_i = \max(x)$, and $x_i \neq \max(x)$.

Case 1 $x_i = \max(x)$.

$$
\begin{aligned}
&|\text{softmax}(cx)_i - \text{softmax}(x)_i| \\
&= \text{softmax}(cx)_i - \text{softmax}(x)_i \\
&\leq 1 - \text{softmax}(x)_i \\
&= 1 - \frac{\exp(x_i)}{\sum_j \exp(x_j)} \\
&\leq 1 - \frac{\exp(\max(x))}{\exp(\max(x)) + (n-1)\exp(\min(x))} \\
&= 1 - \frac{1}{1 + (n-1)\exp(\min(x) - \max(x))} \\
&= 1 - \frac{1}{1 + \exp(\min(x) - \max(x) + d)},
\end{aligned}
$$

for some $d \in \mathbb{R}$. As this has the form of σ,

$$
\begin{aligned}
&|\text{softmax}(cx)_i - \text{softmax}(x)_i| \\
&= 1 - \sigma(\Theta(\|x\|)) = \exp(-\Theta(\|x\|)).
\end{aligned}
$$

Case 2 $x_i \neq \max(x)$.

$$
\begin{aligned}
&|\text{softmax}(cx)_i - \text{softmax}(x)_i| \\
&= \text{softmax}(x)_i - \text{softmax}(cx)_i \\
&\leq 1 - \max(\text{softmax}(x)) - 0 \\
&= 1 - \text{softmax}(\max(x)),
\end{aligned}
$$

which is identical to case 1. $\square$

Finally, for completeness, we show that $\tanh$ exhibits the same property. The proof is very similar to sigmoid, following closely from the definition

$$\tanh(x) = \frac{\exp(2x) - 1}{\exp(2x) + 1}.$$

Lemma 8 *The scaling error for* $\tanh$ *vanishes exponentially in the preactivation magnitude, i.e. for all $c \geq 1$,*

$$|\tanh(cx) - \tanh(x)| \leq \exp(-\Theta(|x|)).$$

Proof.

$$
\begin{aligned}
|\tanh(cx) - \tanh(x)| &\leq |1 - \tanh(x)| \\
&= 1 - \tanh(|x|) \\
&= 1 - \frac{\exp(2|x|) - 1}{\exp(2|x|) + 1} \\
&= \frac{\exp(2|x|) + 1 - \exp(2|x|) + 1}{\exp(2|x|) + 1} \\
&= \frac{2}{\exp(2|x|) + 1} \\
&= \exp(-\Theta(|x|)).
\end{aligned}
$$

$\square$

Thus, applying these functions to a homogeneous input produces an output that is approximately scale-invariant in the parameters θ. Thus, these functions act similarly to layer norm, which maps homogeneous input to scale-invariant output. But what happens if the input is *approximately* homogeneous, rather than strictly homogeneous? In this case, we show that the output is approximately scale-invariant assuming $\|\theta\|$ is sufficiently large.

Proposition 3 *Let $f(x; \theta)$ be approximately k-homogeneous in θ. Then the following functions are approximately scale-invariant in θ:*

$$
\begin{aligned}
g_\sigma &= \sigma \circ f \\
g_{\text{softmax}} &= \text{softmax} \circ f \\
g_{\tanh} &= \tanh \circ f.
\end{aligned}
$$

Proof. If $f(x; \theta)$ is approximately k-homogeneous, then $f(x; c\theta) = c^k f(x; \theta) + \epsilon$ where $\|\epsilon\| \leq \exp(-O(\|\theta\|))$. Crucially, since ϵ vanishes for large norm, there is some ρ where, for all θ such that $\rho < \|\theta\|$:

$$
\begin{aligned}
\text{sgn}\left(c^k f(x; \theta) + \epsilon\right) &= \text{sgn}\left(c^k f(x; \theta)\right) \\
\arg\max\left(c^k f(x; \theta) + \epsilon\right) &= \arg\max\left(c^k f(x; \theta)\right).
\end{aligned}
$$

Therefore, for θ such that $\|\theta\| > \rho$, the bounds used in Lem. 6, Lem. 7, and Lem. 8 hold for approximately homogeneous f. Thus, we can conclude that the output is approximately scale-invariant. □

D   Transformers

We introduce the notation $\sim k$-homogeneous to mean approximately k-homogeneous. In this section, we show that the transformer encoder is ~ 1-homogeneous. A transformer Vaswani et al. (2017) is made up of three main components: an embedding layer, self attention sublayers, and feedforward sublayers. Since the embedding layer is just a matrix multiplication, it is a 1-homogeneous function of the input. Assuming the self attention and feed-forward sublayers have no bias terms, we show that they approximate functions preserving approximate 1-homogeneity. As the full network is an initial embedding layer followed by these sublayers, the final output is ~ 1-homogeneous. In the main paper, we discuss the connection between homogeneity and norm growth.

We base our analysis on the HuggingFace implementation[16] of BERT (Wolf et al., 2019). To aid analysis, we make some simplifying assumptions, which are discussed along with the definitions. We later show empirically that homogeneity for the unsimplified versions is similar.

D.1   Transformer Definition

The transformer encoder is a cascade of alternating *multi-head self-attention* sublayers and *feedforward* sublayers. Each multi-head self-attention sublayer can be further broken down as an aggregation of *self-attention* heads. Let $\text{LN}(\cdot)$ denote a layer norm followed by a learned affine transformation. Here we will consider the pre-norm transformer variant (Xiong et al., 2020), meaning that LN comes before the residual connection wherever it appears.[17] We will also assume that there are no biases, making all affine transformations into strict linear transformations.

Definition 4 (Self-attention head) Given parameters W^k, W^q, W^v and input $X \in \mathbb{R}^{Tn}$, we define

[17]The post-norm transformer applies these operations in the opposite order.

a *self-attention head* attn as

$$K = W^k X$$
$$Q = W^q X$$
$$V = W^v X$$
$$A = \text{softmax}(QK^\top/\sqrt{d_k})$$
$$H = AV,$$

where H is the output tensor.

The multi-head self-attention sublayer computes several attention heads in parallel and aggregates them into a single sequence of vectors.

Definition 5 (Multi-head self-attention sublayer) Let $X \in \mathbb{R}^{Tn}$ be the input. We now define the *k-multi-head self-attention sublayer* MSA_k. First, we compute k self-attention heads in parallel to produce $H_1, \cdots, H_k$. We then concatenate these along the feature axis to form H, and compute the sublayer output Y as

$$\text{MSA}_k(X) = \text{LN}(WH) + X.$$

Finally, the linear sublayer is the other component of the transformer.

Definition 6 (Feedforward sublayer) Let $X \in \mathbb{R}^{Tn}$ be the input. We compute the *feedforward sublayer* FF according to

$$\text{FF}(X) = \text{LN}(W^f \text{ReLU}(W^i X)) + X.$$

D.2   Results

Proposition 4 *If X is ~ 1-homogeneous in parameters θ, then* $\text{attn}(X; W^k, W^q, W^v)$ *is ~ 1-homogeneous in the concatenation of* θ, W^k, W^q, W^v.

Proof. Consider a self-attention layer receiving a ~ 1-homogeneous input matrix $X \in \mathbb{R}^{Tn}$ where T is the sequence length. Using the homogeneity rule for multiplication, K, Q, V are each ~ 2-homogeneous, as homogeneity is additive over multiplication. By the same argument, $QK^\top$ is ~ 4-homogeneous. In Prop. 3, we show that if the input to softmax is approximately homogeneous, then the output is approximately scale-invariant. Thus, A is approximately 0-homogeneous. Then AV is ~ 1-homogeneous. □

We show that the multi-head component that aggregates multiple heads into a shared representation also preserves approximate 1-homogeneity.

Proposition 5 *If X is ~ 1-homogeneous in parameters θ, then* MSA *is ~ 1-homogeneous in the full parameters.*

Proof. Since Wh is ~ 2-homogeneous, $\text{LN}(WH)$ is ~ 1-homogeneous. The input X is also ~ 1-homogeneous by assumption, meaning that the sum is also ~ 1-homogeneous. □

Finally, we turn to analyzing the feedforward sublayer of the transformer.

Proposition 6 *If X is ~ 1-homogeneous, then* $\text{FF}(X; W^f, W^i)$ *is ~ 1-homogeneous in the full parameters.*

Proof. Multiplying by each W increases approximate homogeneity by 1, and ReLU preserves approximate homogeneity. So the input to LN is ~ 3-homogeneous. Thus, its output is ~ 1-homogeneous, and adding X preserves approximate 1-homogeneity. □

Together, these results suggest that the pre-norm transformer output is ~ 1-homogeneous, assuming its input is ~ 1-homogeneous. This precondition for the input holds in the "base case" of standard embeddings. By induction, we can imagine the output of a biasless pre-norm transformer encoder of any depth to be ~ 1-homogeneous.

Interestingly, the homogeneity arguments do not work out if we instead consider the post-norm transformer architecture (Xiong et al., 2020).

E Sum Derivation

E.1 Aligned Case

Assume that $\|\theta_t\| \approx 0$. Then,

$$\|\theta_t\| \propto \int \eta_t \|\theta_t\| \mathrm{d}t$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \|\theta_t\| \propto \eta_t \|\theta_t\|$$

$$\frac{\mathrm{d}\|\theta_t\|}{\|\theta_t\|} \propto \eta_t \mathrm{d}t$$

$$\log \|\theta_t\| \propto \int \eta_t \mathrm{d}t$$

$$\|\theta_t\| \propto \exp\left(\int \eta_t \mathrm{d}t\right).$$

Plugging in $\eta_t = 1/\sqrt{t}$, we get $\|\theta_t\| \propto \exp(\sqrt{t})$.

E.2 Misaligned Case

First, we derive the sum approximation for $\|\theta_t\|$. We start with the fact that $\theta_{t+1} = \theta_t + \delta_t$ and misalignment, i.e., $\theta_t^\top \cdot \delta_t = 0$.

$$\begin{aligned}\|\theta_{t+1}\|^2 &= (\theta_t + \delta_t) \cdot (\theta_t + \delta_t) \\ &= \|\theta_t\|^2 + \theta_t^\top \delta_t + \|\delta_t\|^2 \\ &= \|\theta_t\|^2 + \|\delta_t\|^2 \\ &= \|\theta_0\|^2 + \sum_{i=0}^{t} \|\delta_i\|^2.\end{aligned}$$

Taking the square root of both sides, $\|\theta_t\|$ is roughly proportional to $\sum_{i=0}^{t} \|\delta_i\|^2$.

Next, we show how to solve the integral, similarly to §E.1.

$$\|\theta_t\|^2 \propto \int \|\delta_t\|^2 \mathrm{d}t$$

$$\|\theta_t\|^2 \propto \int \eta_t^2 \|\theta_t\|^2 \mathrm{d}t$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \|\theta_t\|^2 \propto \eta_t^2 \|\theta_t\|^2$$

$$\frac{\mathrm{d}\|\theta_t\|^2}{\|\theta_t\|^2} \propto \eta_t^2 \mathrm{d}t$$

$$\log \|\theta_t\|^2 \propto \int \eta_t^2 \mathrm{d}t.$$

Now, we plug in the $\eta_t = 1/\sqrt{t}$ learning rate:

$$\begin{aligned}\log \|\theta_t\|^2 &\propto \int (1/\sqrt{t})^2 \mathrm{d}t \\ &\propto \int \frac{\mathrm{d}t}{t} \\ &\propto \log t.\end{aligned}$$

So, in conclusion: $\|\theta_t\| \propto \sqrt{t}$.

F Weight Decay

Weight decay regularizes the loss by the squared ℓ_2 norm, modulated by a decay factor λ. For GD, this can be written

$$\delta_t = -\eta_t \nabla_{\theta_t} L - \lambda \theta_t. \tag{3}$$

Intuitively, the new term $-\lambda \theta_t$ will influence each step to point towards 0. Thus, large values of λ might intuitively be expected to hinder or prevent norm growth. While we leave developing a more complete theoretical story to future work, here we empirically investigate the interplay of a constant
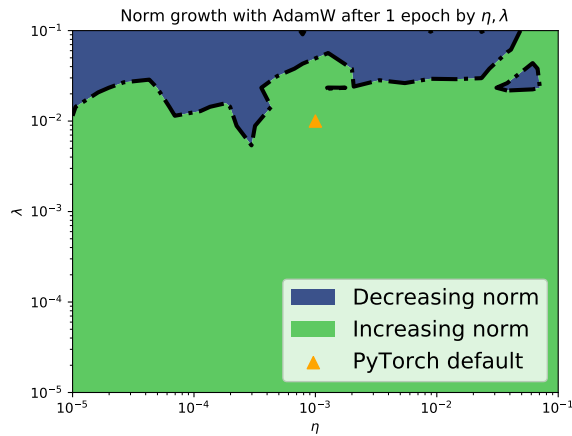
Figure 6: Norm growth over the first epoch, varying η, λ. The triangle shows the default AdamW hyperparameters in PyTorch.

learning rate η and weight decay λ by training a variety of transformer language models on Wikitext-2 for 1 epoch.[18] We use the AdamW (Loshchilov and Hutter, 2017) optimizer, varying λ and η across a range of common values, keeping all other hyperparameters constant. Fig. 6 visualizes the phase transition for norm growth as a function of λ, η. The norm growth behavior seems to largely depend on weight decay, with a threshold for λ lying between 0.01 and 0.001. While the trend likely depends on the optimizer, we can infer for AdamW at least that norm growth is probable when $\lambda = 0.01$, which is a common choice, e.g., reflecting default settings in PyTorch. Thus, while large values of λ will indeed hinder norm growth, we find preliminary empirical evidence that standard choices (~ 0.01) do not prevent it.

---

[18] 1 epoch is chosen because of the computational cost of running this experiment over a large grid. In §3, we found that growth continued beyond 1 epoch using the default AdamW settings.