TAMING TRANSFORMER WITHOUT USING LEARNING RATE WARMUP

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling Transformer to a large scale without using some technical tricks such as learning rate warump and an obviously lower learning rate, is an extremely challenging task, and is increasingly gaining more attention. In this paper, we provide a theoretical analysis for training Transformer and reveal a key problem behind the model crash phenomenon in the training, *i.e.*, the *spectral energy concentration* of ${\boldsymbol{W}_q}^\top \boldsymbol{W}_k$ (where $\boldsymbol{W}_q$ and $\boldsymbol{W}_k$ are the projection matrices for query and key in Transformer), which is the reason for a malignant entropy collapse. To remedy this problem, motivated by *Weyl's Inequality*, we present a novel optimization strategy—making weight updating in successive steps smooth, that is, if the ratio $\frac{\sigma_1(\nabla \boldsymbol{W}_t)}{\sigma_1(\boldsymbol{W}_{t-1})}$ is larger than a threshold, where $\nabla \boldsymbol{W}_t$ is the updating quantity in step t, we will automatically bound the learning rate to a weighted multiply of $\frac{\sigma_1(\boldsymbol{W}_{t-1})}{\sigma_1(\nabla \boldsymbol{W}_t)}$. Our optimization strategy is able to prevent the rapid spectral energy concentration to only a few directions, and thus is able to avoid the malignant entropy collapse that will trigger the model crash. We conduct extensive experiments using ViT, Swin-Transformer and GPT, showing that our optimization strategy can effectively and stably train these (Transformer) models without using learning rate warmup.

"Nothing in life is to be feared. It is only to be understood."
— Marie Curie

1 INTRODUCTION

Transformer (Vaswani et al., 2017) has revolutionized various domains of artificial intelligence, including natural language processing (Radford et al., 2018; 2019; Brown et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023; Dubey et al., 2024) and computer vision (Dosovitskiy et al., 2020; Liu et al., 2021) and many more applications (Radford et al., 2021; Ramesh et al., 2021; Peebles & Xie, 2023), owning to their ability to capture long-range dependencies through self-attention mechanisms. However, despite their widespread application and empirical success, training deep Transformer models remains quite challenging. Practitioners frequently encounter variant issues, such as gradient explosion (Qi et al., 2023b), rank collapse (Dong et al., 2021), entropy collapse (Zhai et al., 2023) and general training instability (Kim et al., 2021; Qi et al., 2023b), especially during the initial stages of the training.

To address these challenges, researchers have proposed various modifications to the original Transformer architecture, including altering the placement of the Layer Normalization (Wang et al., 2019; Xiong et al., 2020) (*e.g.*, pre-LN vs. post-LN schemes), carefully conditioning the residual connections Bachlechner et al. (2021), and QKNorm (Henry et al., 2020; Dehghani et al., 2023) for self-attention module. Similarly, DeepNet (Wang et al., 2022) introduces a new normalization function to modify the residual connection in Transformer. ReZero (Bachlechner et al., 2021) introduces a learnable residual scalar parameter for the residual shortcut, and requiring to initiate it to 0 at the start time of training. More recent approaches (Kim et al., 2021; Qi et al., 2023a) have focused on examining and enforcing Lipschitz continuity properties of Transformer components, which can provide insights into the network behavior and the training stability. *Al-*

though there are a few works (Bachlechner et al., 2021; Qi et al., 2023a) that can avoid using learning rate warmup to train the Transformer successfully, all of them require significant modifications of the network architecture.

Learning rate warmup (Loshchilov & Hutter, 2016) seems to be a must-have technology for standard optimizers (Robbins & Monro, 1951; Duchi et al., 2011; Kingma & Ba, 2014; Loshchilov & Hutter, 2019) in some popular large Transformer models (Radford et al., 2018; 2019; Brown et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023). Without the learning rate warmup stage, the Transformer training will prone to diverge.

Although it is usual to train a Transformer by modifying the network structure as mentioned above or using the learning rate warmup, two natural and interesting questions remain:

1. *What are the training dynamics of a Transformer model when its training fails or successes?*
2. *Can we successfully tame an arbitrary Transformer without changing its network structure or without using learning rate warmup?*

This paper aims to answer these questions. To answer the first question, we visualize the training processes of three types of Transformers, construct 15 or 13 quantities of parameters and activations, and visualize their change trajectories along with the training process and the changes of the attention maps. By doing so, we observe that the model crash is accompanied by a weird phenomenon that the entropy of the attention map is almost 0 and the spectral norm of ${\boldsymbol{W}_q}^\top \boldsymbol{W}_k$ increases to a very large value. By conducting mathematical analysis for the Transformer training, we identify that the spectral energy concentration ${\boldsymbol{W}_q}^\top \boldsymbol{W}_k$ is the key problem to leading the model crash. To answer the second question, motivated by Weyl' Inequality, we present a novel optimization strategy—making weight updating smooth, and verify empirically that our optimization strategy can prevent the rapid spectral energy concentration and thus achieving a stable convergence in training.

Paper Contributions. The contributions of the paper are highlighted as follows.

- We visualize the training dynamics of Transformers that train successfully or unsuccessfully and summarize two important observations from unsuccessful training that: a) the rank of the attention map matrix tends to very low and the entropy of attention probability matrix tends to 0; and b) $\sigma_1({\boldsymbol{W}_q}^\top \boldsymbol{W}_k)$ increases rapidly to a very large value.
- We present theoretical analysis for the Transformer training, finding that the Jacobian matrix $\frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}({\boldsymbol{W}_q}^\top \boldsymbol{W}_k)} = \boldsymbol{X}^\top \otimes \boldsymbol{X}^\top$, where $\boldsymbol{P} = \boldsymbol{X}^\top {\boldsymbol{W}_q}^\top \boldsymbol{W}_k \boldsymbol{X}$. It implies that the gradient of ${\boldsymbol{W}_q}^\top \boldsymbol{W}_k$ is largely dominated by the rank of $\boldsymbol{X}^\top \otimes \boldsymbol{X}^\top$.
- We reveal that the Spectral Energy Concentration (SEC) of $\boldsymbol{W}_q^\top \boldsymbol{W}_k$ makes the attention map matrix to be sparse yet low-rank and it is the inner reason leading to model crash.
- Motivated by Weyl's inequality, We introduce a novel strategy to address the problem of spectral energy concentration of ${\boldsymbol{W}_q}^\top \boldsymbol{W}_k$ by controlling the rapid growth of singular values, and verify that our strategy leads to a stable training process.

2 Preliminaries

We first review some notions or algorithm, *i.e.*, matrix norm, power iteration, Adam optimizer that will be used in this paper.

Matrix norm. Given a matrix $\boldsymbol{W}$, its p-norm is a non-negative real number denoted $\|\boldsymbol{W}\|_p$. A classical definition of matrix norm is induced by vector p-norm, it is defined as: $\|\boldsymbol{W}\|_p = \sup_{\boldsymbol{x}\neq 0} \frac{\|\boldsymbol{W}\boldsymbol{x}\|_p}{\|\boldsymbol{x}\|_p}$. When $p = 2$, the induced matrix norm is the *spectral norm.* The spectral norm of a matrix $\boldsymbol{W}$ is defined as the largest singular value of $\boldsymbol{W}$. This can also be expressed as the square root of the largest eigenvalue of the Gram matrix $\boldsymbol{W}^\top \boldsymbol{W}$. The spectral norm of a matrix $\boldsymbol{W}$ can be calculated as: $\|\boldsymbol{W}\|_2 = \max_{\boldsymbol{x}\in S^{n-1}} \|\boldsymbol{A}\boldsymbol{x}\|_2 = \sqrt{\lambda_{\max}(\boldsymbol{W}^\top \boldsymbol{W})} = \sigma_1(\boldsymbol{W})$, where $\sigma_1(\boldsymbol{W})$ represents the largest singular value of matrix $\boldsymbol{W}$, which is also be denoted by $\sigma_{\max}(\boldsymbol{W})$, S^{n-1} denotes a unit sphere in $\mathcal{R}^n$ with radius 1.

Power iteration to compute matrix norm. The power iteration algorithm starts with a vector $\boldsymbol{x}_0$ (to speedup the iteration, we usually use ℓ_2-norm to normalize $\boldsymbol{x}_0$). The entire iteration process is as follows: $\boldsymbol{x}_{k+1} = \frac{\boldsymbol{W}\boldsymbol{x}_k}{\|\boldsymbol{W}\boldsymbol{x}_k\|_2}$ for $k = 0, \cdots, K-1$. At every iteration, $\boldsymbol{x}_k$ is multiplied by the matrix $\boldsymbol{W}$ and normalized. The final $\|\boldsymbol{x}_K\|_2$ is returned as the final spectral norm. Usually, it takes 3 to 5 iterations to converge. The overall complexity is low.

Adam Optimizer. Adam optimizer (Kingma & Ba, 2014) is currently the most widely used optimizer for training neural networks, owing to its efficiency and effectiveness. Adam can be simply defined as: $\boldsymbol{M}_t = \beta_1 \boldsymbol{M}_{t-1} + (1-\beta_1)\boldsymbol{G}_t, \quad \boldsymbol{V}_t = \beta_2 \boldsymbol{V}_{t-1} + (1-\beta_2)\boldsymbol{G}_t^2, \quad \boldsymbol{W}_t = \boldsymbol{W}_{t-1} - \alpha_t \boldsymbol{M}_t \oslash \sqrt{\boldsymbol{V}_t + \epsilon}$ where $\boldsymbol{G}_t$ is the gradient at time step t, $\boldsymbol{G}_t^2$ is the element-wise square of $\boldsymbol{G}_t$, and $\oslash$ denotes element-wise division, α_t is the learning rate at time step t, (β_1, β_2) are the first-order and the second-order momentum factors.

3 Taming Transformer requires Revisiting its Training Dynamics

Before starting to analyze the training dynamics of Transformer, we first give some basic notions in Transformer, which includes an attention module, an FFN module and two normalization modules which is used before attention module or FFN module. For the attention module, we usually use multi-head attention that allows the model to jointly attend to information from different representations from different heads. Here, for convenience of definition without losing generality, we only use a single-head attention. To be precise, we define each of them as follows.

$$\begin{aligned} \mathrm{Attn}(\boldsymbol{X}; \boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v, \boldsymbol{W}_o) &= \boldsymbol{W}_o \boldsymbol{W}_v \boldsymbol{X} \,\mathrm{softmax}\left(\frac{\boldsymbol{X}^\top {\boldsymbol{W}_q}^\top \boldsymbol{W}_k \boldsymbol{X}}{\sqrt{d_q}}\right), \\ \mathrm{FFN}(\boldsymbol{x}; \boldsymbol{W}_1, \boldsymbol{W}_2) &= \boldsymbol{W}_2 \,\mathrm{ReLU}(\boldsymbol{W}_1 \boldsymbol{x}), \\ \mathrm{LN}(\boldsymbol{x}) &= \boldsymbol{\gamma} \odot \boldsymbol{z} + \boldsymbol{\beta}, \quad \text{where } \boldsymbol{z} = \frac{\boldsymbol{y}}{\mathrm{std}(\boldsymbol{y})} \text{ and } \quad \boldsymbol{y} = \left(\boldsymbol{I} - \frac{1}{D}\mathbf{1}\mathbf{1}^\top\right)\boldsymbol{x}. \end{aligned}$$

Here, $\boldsymbol{X} \in \mathcal{R}^{d \times n}$, $\boldsymbol{W}_q, \boldsymbol{W}_k \in \mathcal{R}^{d_q \times d}$, $\boldsymbol{W}_v \in \mathcal{R}^{d_v \times d}$, $\boldsymbol{W}_o \in \mathcal{R}^{d \times d}$, $\boldsymbol{W}_1 \in \mathcal{R}^{4d \times d}$, $\boldsymbol{W}_2 \in \mathcal{R}^{d \times 4d}$, $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathcal{R}^d$. Note that only in a single-head definition, $\boldsymbol{W}_o$ can be put before $\boldsymbol{W}_v$, otherwise, it should be after a concatenation operator. For the sake of further discussion, we will define the following notations:

$$\boldsymbol{P} = \boldsymbol{X}^\top {\boldsymbol{W}_q}^\top \boldsymbol{W}_k \boldsymbol{X}, \quad \boldsymbol{A} = \mathrm{softmax}(\frac{\boldsymbol{P}}{\sqrt{d_q}}),$$

where $\boldsymbol{A}$ is the attention map, and $\frac{\boldsymbol{P}}{\sqrt{d_q}}$ is usually called as the logit.

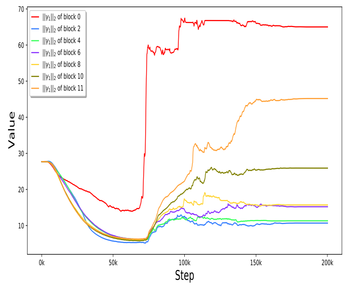
3.1 Visualization: What happens when a Transformer training fails or succeeds

Visualizations is a commonly used effective technology to help us understand why the training of a neural networks work or fail. In particular, what happens when the training of a Transformer collapses? And what happens when its training successes? Visualization of these phenomenons can help us obtain a deeper understanding of the process of training Transformer.

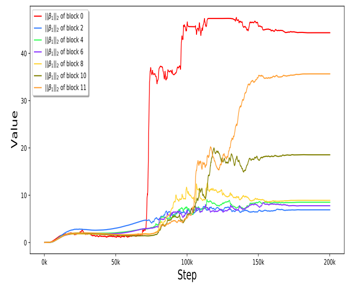
One of the most important aspects of understanding the training of neural networks is the observation of changes in parameters and activations. Since that the parameters or activations and their gradients are basically matrices or vectors, the matrix norm is the best way to look at the statistics of these quantities. In this paper, for a Transformer training, we summarize the following **15 terms** to watch:

$$\begin{aligned} &\sigma_1(\boldsymbol{W}_q), \quad \sigma_1(\boldsymbol{W}_k), \quad \sigma_1(\boldsymbol{W}_v), \quad \sigma_1(\boldsymbol{W}_o), \quad \sigma_1(\boldsymbol{W}_1), \quad \sigma_1(\boldsymbol{W}_2), \quad \|\boldsymbol{\gamma}_1\|_2, \quad \|\boldsymbol{\beta}_1\|_2, \quad \|\boldsymbol{\gamma}_2\|_2, \quad \|\boldsymbol{\beta}_2\|_2, \\ &\sigma_1({\boldsymbol{W}_q}^\top \boldsymbol{W}_k), \quad \sigma_1(\boldsymbol{W}_o \boldsymbol{W}_v), \quad \sigma_1(\boldsymbol{W}_2 \boldsymbol{W}_1), \quad \|\boldsymbol{x}\|_2, \quad \left\|\frac{\partial L}{\partial \boldsymbol{x}}\right\|_2, \end{aligned} \tag{1}$$

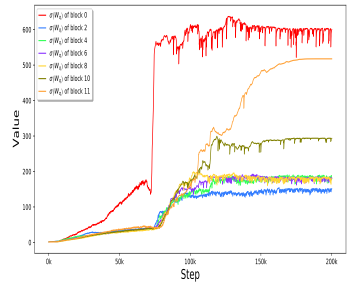
where $\boldsymbol{\beta}_1, \boldsymbol{\gamma}_1$ and $\boldsymbol{\beta}_2, \boldsymbol{\gamma}_2$ denote the parameters of the first and the second LayerNorm. When RMSNorm (Zhang & Sennrich, 2019) is used, only 13 terms will be analyzed since that it does



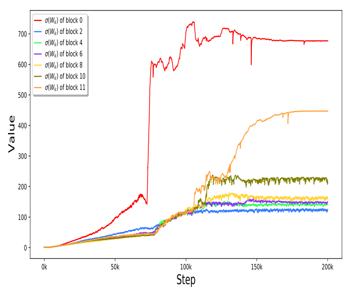
(a) $\|\boldsymbol{\gamma_1}\|_2$



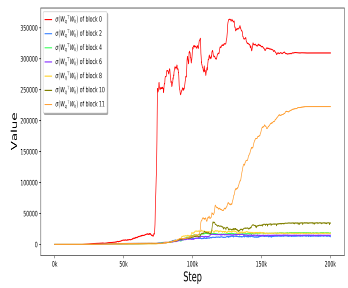
(b) $\|\boldsymbol{\beta_1}\|_2$



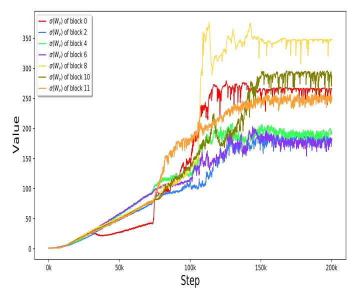
(c) $\sigma_1\left(\boldsymbol{W}_q\right)$



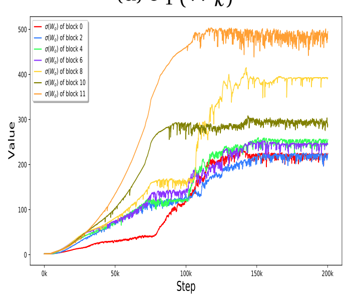
(d) $\sigma_1\left(\boldsymbol{W}_k\right)$



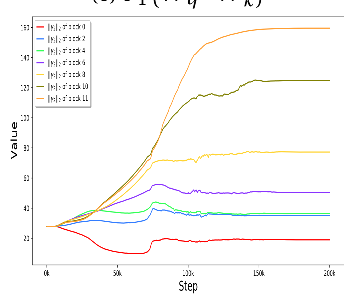
(e) $\sigma_1\left(\boldsymbol{W}_q{}^\top \boldsymbol{W}_k\right)$



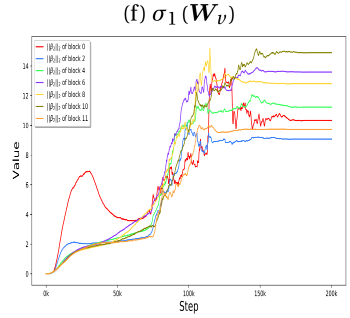
(f) $\sigma_1\left(\boldsymbol{W}_v\right)$



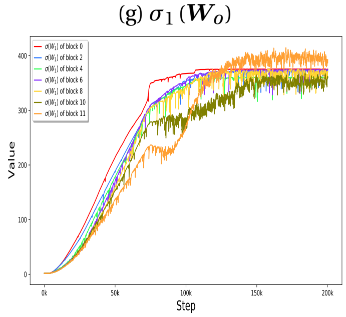
(g) $\sigma_1\left(\boldsymbol{W}_o\right)$



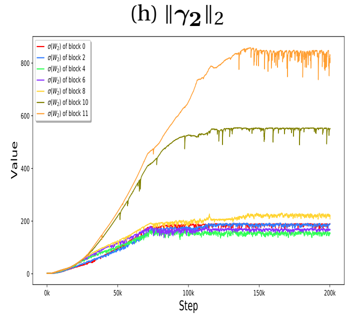
(h) $\|\boldsymbol{\gamma_2}\|_2$



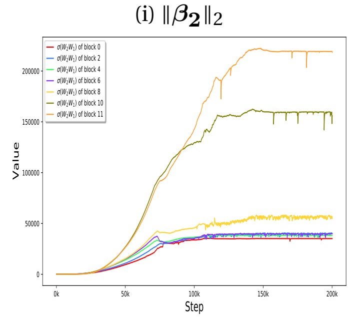
(i) $\|\boldsymbol{\beta_2}\|_2$



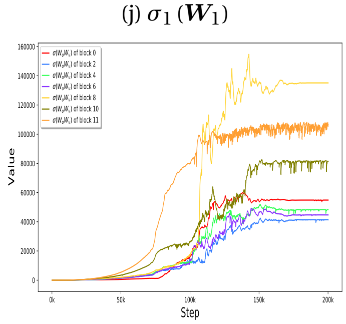
(j) $\sigma_1\left(\boldsymbol{W}_1\right)$



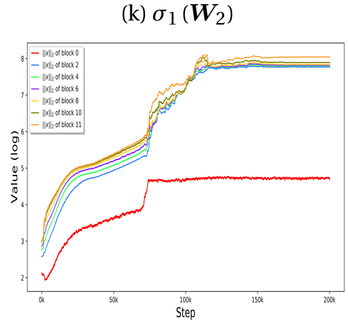
(k) $\sigma_1\left(\boldsymbol{W}_2\right)$



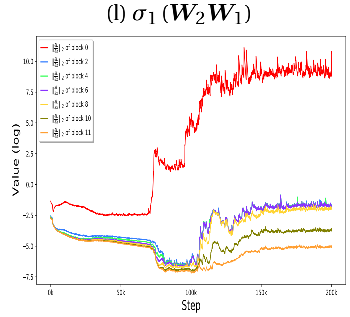
(l) $\sigma_1\left(\boldsymbol{W}_2\boldsymbol{W}_1\right)$



(m) $\sigma_1\left(\boldsymbol{W}_o\boldsymbol{W}_v\right)$



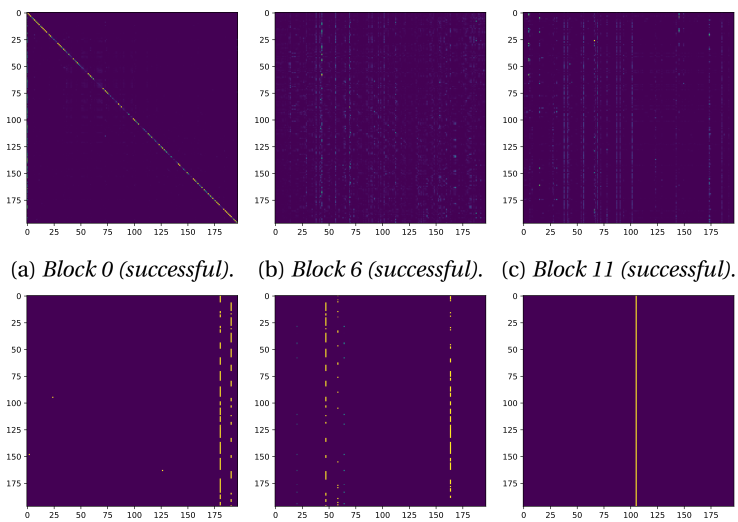
(n) $\|\boldsymbol{x}\|_2$



(o) $\|\frac{\partial L}{\partial \boldsymbol{x}}\|_2$

FIGURE 1: Training dynamics of a failure ViT. This figure shows how the values of these 15 items as shown in Equation 1 change as the number of training steps increases. Please pay more attention to subfigures (a)-(e).

not have $\boldsymbol{\beta}$. For the weight matrix, we use the spectral norm which is defined by the maximum singular value. For a vector, we use its ℓ_2 norm.



(a) *Block 0 (successful).* (b) *Block 6 (successful).* (c) *Block 11 (successful).*

(d) *Block 0 (crashed).* (e) *Block 6 in (crashed).* (f) *Block 11 (crashed).*

FIGURE 2: Visualization of the dynamics process of attention map in different training steps for a successful and a crashed ViT-Base model, respectively. *Please click the images to play the flash. Best viewed with Acrobat Reader.*

To ensure that the phenomena we observed can generalize well, we visualized them on both ViT (Dosovitskiy et al., 2020) and GPT (Radford et al., 2018). ViT is a pure encoder architectures, and GPT is a pure decoder architecture. In this section, due to the limitations of paper space, we will only visualize ViT-Base, and put more visualization results into the Appendix H. For the ViT implementation, we use Timm Wightman (2019), in which "timm" library provides rich model architectures of many pre-trained image models in PyTorch. For the GPT implementation, we use nanoGPT, which uses LayerNorm without bias term, and thus only adopt 13 terms instead of 15 terms in ViT.

To achieve a successful ViT training, we use a long learning rate warmup. For instance, we use 60 epochs of warmup and the whole training process takes 150 epochs. To obtain the dynamics of a failure training of ViT, we do not use warmup.

Figure 1 visualizes a failure training process of a ViT-Base model. The visualized model includes 12 blocks with index from 0 to 11. In Figure 1, we visualize the weight matrices in blocks [0, 2, 4, 6, 8, 10, 11]. Block 11 is the last block. For the features $\boldsymbol{x}$ and the gradients $\frac{\partial l}{\partial \boldsymbol{x}}$, we hook the input features that enter into the corresponding blocks. Figure 8 in the Appendix I visualizes a successful training process of a ViT-L model. Meanwhile, in Figure 2, we visualize the dynamic process of attention map as the number of training steps increases for a successful and failure ViT-Base model. In different time steps, we visualize the attention map of the same image.

We observe the following phenomena from Figures 1 and 2.

- As shown in Figure 1, at the beginning, the maximum singular value of $\sigma_1(\boldsymbol{W}_q{}^\top \boldsymbol{W}_k)$ gradually increases, and at a certain point, the maximum singular value suddenly and rapidly increases to a very large value (around 200,000),.that is, the loss diverge at this time. However, for a successful training process, $\sigma_1(\boldsymbol{W}_q{}^\top \boldsymbol{W}_k)$ gradually increases to a medium value as shown in Figure 8 and then vibrate around that value.
- As shown in Figure 2, in a failure training process of Transformer, the attention map gradually becomes sparse and low-rank, and finally collapses to a very sparse and low-rank mode. In a collapsed model, the entropy of the attention map is 0. However, in a successful training process, the attention map is not too sparse and have a medium rank.
- The normalization layers exhibit a huge difference: the $\boldsymbol{\gamma_1}$ and $\boldsymbol{\beta_1}$ in a successful ViT training process are very smooth, but they change a lot in an unsuccessful case.
- The ranges of the activation and the gradients are very large in a crashed model, and the gradients in different blocks vary much larger than that in a successful model.

Remark. We summarize that the successful training and the unsuccessful training of Transformer exhibit significant differences between their $\sigma_1(\boldsymbol{W}_q{}^\top \boldsymbol{W}_k)$, their normalization parameters γ and $\boldsymbol{\beta}$, and their activations $\boldsymbol{x}$ and gradients $\frac{\partial L}{\partial \boldsymbol{x}}$.

3.2 Theoretical Analysis: Matrix Calculus of Transformer

To understanding the training dynamics of Transformer, we should investigate the process of back-propagation (Rumelhart et al., 1986; LeCun et al., 2002; 1989; 1998). In the attention mechanism, however, the input and the output are both matrices, we cannot directly use vector calculus. Instead, we need to use *Vectorization* [1] (Graham, 2018; Petersen et al., 2008) and *Kronecker Product* [2] (Graham, 2018; Petersen et al., 2008). In matrix calculus, the vectorization of a matrix is a linear transformation that converts a matrix into a vector. Specifically, the vectorization of a matrix $\boldsymbol{M} \in \mathcal{R}^{m \times n}$, denoted as $\text{vec}(\boldsymbol{M})$, is a column vector, obtained by an ordered stacking of the columns of the matrix $\boldsymbol{M}$, *i.e.*, $\text{vec}(\boldsymbol{M}) \in \mathcal{R}^{mn}$. For example, for a 2×3 matrix $\boldsymbol{M} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$, the vectorization of $\boldsymbol{M}$ is $\text{vec}(\boldsymbol{M}) = [a\ d\ b\ e\ c\ f]^\top$. For the attention module, we have the following proposition about the Jacobian matrix of the output $\boldsymbol{P}$ with respect to the input $\boldsymbol{X}$ and the parameters.

Proposition 1 (Matrix Calculus for Self-Attention)
Let $\boldsymbol{P} = \boldsymbol{X}^\top \boldsymbol{W}_q{}^\top \boldsymbol{W}_k \boldsymbol{X}$, where $\boldsymbol{X} \in \mathcal{R}^{d \times n}, \boldsymbol{W}_q \in \mathcal{R}^{d_q \times d}, \boldsymbol{W}_k \in \mathcal{R}^{d_q \times d}$, according to vectorization and matrix calculus, we have the following derivations:

$$\frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{W}_q{}^\top \boldsymbol{W}_k)} = \boldsymbol{X}^\top \otimes \boldsymbol{X}^\top, \quad \frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{X})} = (\boldsymbol{X}^\top \boldsymbol{W}_q{}^\top \boldsymbol{W}_k \otimes \boldsymbol{I}_n)\boldsymbol{K} + (\boldsymbol{I}_n \otimes \boldsymbol{X}^\top \boldsymbol{W}_q{}^\top \boldsymbol{W}_k), \tag{2}$$

$$\frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{W}_q^\top)} = \boldsymbol{X}^\top \otimes (\boldsymbol{W}_k \boldsymbol{X})^\top, \quad \frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{W}_k)} = \boldsymbol{X}^\top \otimes (\boldsymbol{W}_q \boldsymbol{X})^\top, \tag{3}$$

where $\otimes$ denotes Kronecker product, $\boldsymbol{I}_n \in \mathcal{R}^{n \times n}$ denotes an identity matrix with shape $n \times n$, $\boldsymbol{K}$ is the commutation matrix, which depends on the dimensions of $\boldsymbol{X}$. Since $\boldsymbol{X} \in \mathcal{R}^{d \times n}$, then we know $\boldsymbol{K} \in \mathcal{R}^{nd \times nd}$. The commutation matrix $\boldsymbol{K}$ has the property that $\text{vec}(\boldsymbol{X}^\top) = \boldsymbol{K}\,\text{vec}(\boldsymbol{X})$ for any matrix $\boldsymbol{X}$.

In Appendices A and B, we supply some elementary background information for vectorization and Kronecker product and the derivation of Jacobian matrix for a single-head attention.

We have the following observations from Proposition 1.

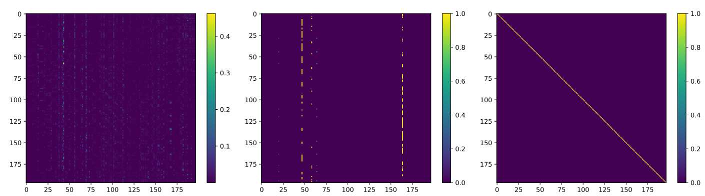
- We have $\frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{W}_q{}^\top \boldsymbol{W}_k)} = \boldsymbol{X}^\top \otimes \boldsymbol{X}^\top$ in Equation 2, and we know about the Kronecker product that $\text{rank}(\boldsymbol{X}^\top \otimes \boldsymbol{X}^\top) = \text{rank}(\boldsymbol{X}^\top)^2$, which implies that if $\boldsymbol{X}$ has a very low rank, then $\frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{W}_q{}^\top \boldsymbol{W}_k)}$ will also have a very low rank. Note that $\boldsymbol{X}$ having a low rank means the features across different timestep are highly correlated or coherent. If all $\boldsymbol{x}_i$ in $\boldsymbol{X}$ collapses to a single point, then $\boldsymbol{X}^\top \otimes \boldsymbol{X}^\top$ will only have a large singular value, and the rest are 0.
- the Jacobian matrix $\frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{X})}$ in Equation 2, is in direct proportion to $\boldsymbol{X}$ and $\boldsymbol{W}_q{}^\top \boldsymbol{W}_k$. If the spectral norm $\sigma_1(\boldsymbol{W}_q{}^\top \boldsymbol{W}_k)$ is very large, it implies that the gradient $\frac{\partial L}{\partial \boldsymbol{X}}$ will more likely to be magnified a lot.
- Equation 3 suggests that changes in the query weights $\boldsymbol{W}_q$ are related to both the input $\boldsymbol{X}$ and the key representation $\boldsymbol{W}_k \boldsymbol{X}$. Equation 3 suggests that changes in the key weights $\boldsymbol{W}_k$ are related to both the input $\boldsymbol{X}$ and the query representation $\boldsymbol{W}_q \boldsymbol{X}$.
- All these relationships are interconnected, with changes in one variable potentially affecting the others. For instance, if $\boldsymbol{W}_k$ increases fast, then according to Equation 3, $\frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{W}_q^\top)}$ will more likely to be very large. In this way, $\boldsymbol{W}_q$ will likely to increase fast.

[1] https://en.wikipedia.org/wiki/Vectorization_(mathematics)
[2] https://en.wikipedia.org/wiki/Kronecker_product

3.3 A Key Problem Occurs in Model Crash: Spectral Energy Concentration

Before we reveal the key problem in model crash, let us first discuss the concept so-called entropy collapse.

Two Entropy Collapse Modes. In experiments, we observe two types of attention entropy collapse modes. Note that when attention collapse happens, the attention map tends to a sparse matrix (*i.e.*, there are a few dominate nonzero attention coefficients), and thus the entropy of the attention map is vanishing. To be more specific, when the attention map is sparse but not low-rank, we call it a *benign collapse*; whereas if the attention map is sparse yet low-rank,[3] we call it a *malignant collapse*. When benign collapse occurs, the attention map is shown in the right panel of Figure 3 that, there is almost an identity matrix. In this way, the diagonal elements are almost 1, and the non-diagonal elements have values around 0. Unfortunately, when malignant collapse happens, the attention probability matrix will become to a sparse yet low-rank matrix as shown in middle panel of Figure 3. Furthermore, we observe that the distribution of the spectral energy of $\boldsymbol{W}_q{}^\top \boldsymbol{W}_k$ for the benign collapse is relatively uniform; whereas the spectral energy of the attention matrix for the malignant collapse tends to concentrate on a few dominate singular values.



Normal attention. *Malignant collapse.* *Benign collapse.*

FIGURE 3: Three modes of attention maps. The left panel shows a normal attention map. The middle panel shows a classical attention map when the model crashes, for which the entropy is almost 0. The right panel shows an attention map from a normal model training while its entropy is almost 0.

By analyzing the matrix $\boldsymbol{W}_q{}^\top \boldsymbol{W}_k$ in the benign mode when it happens in the experiments, we find that it has the following property: $\boldsymbol{W}_q{}^\top \boldsymbol{W}_k$ is usually a non-symmetric positive quasi-definite square matrix (see Appendix O for details). In a benign attention mode, the self-attention layer degenerates into a linear projection layer because $\boldsymbol{Y} = \boldsymbol{W}_v \boldsymbol{X} \boldsymbol{A} \approx \boldsymbol{W}_v \boldsymbol{X} \boldsymbol{I} = \boldsymbol{W}_v \boldsymbol{X}$. We give an intuitive analysis in Appendix C.

When the malignant mode happens, the model will usually crash. We identify that the key problem is a phenomenon called *spectral energy concentration* (*SEC*). Before we present our theorem about SEC, let us first introduce an index to quantify *SEC*. Recall that $\boldsymbol{W}_q, \boldsymbol{W}_k \in \mathcal{R}^{d_q \times d}$, where $d_q < d$. We have that $\boldsymbol{W}_q{}^\top \boldsymbol{W}_k \in \mathcal{R}^{d \times d}$, but its rank is less or equal than d_q. Precisely, we define a *SEC index* as follows:

$$\mathrm{SEC}(d_q, s) = \frac{\sum_{i=1}^{s} \sigma_i^2(\boldsymbol{W}_q{}^\top \boldsymbol{W}_k)}{\sum_{i=1}^{d_q} \sigma_i^2(\boldsymbol{W}_q{}^\top \boldsymbol{W}_k)}, \tag{4}$$

where d_q is the head dimension, s is an integer which is not greater than d_q. For instance, if we have $d_q = 64$ and $s = 4$, and if at this time, $\mathrm{SEC}(64, 4) > 99\%$, we could say the spectral energy of $\boldsymbol{W}_q{}^\top \boldsymbol{W}_k$ highly concentrates on only four dominant singular values.

To be precise, we have the following theorem for the reason to cause a malignant collapse.

Theorem 1 (Malignant Entropy Collapse)
Malignant entropy collapse happens (i.e., the attention map $\boldsymbol{A}$ becomes a sparse yet low-rank) with high probability, when $\boldsymbol{W}_q{}^\top \boldsymbol{W}_k$ satisfies the following two conditions simultaneously:

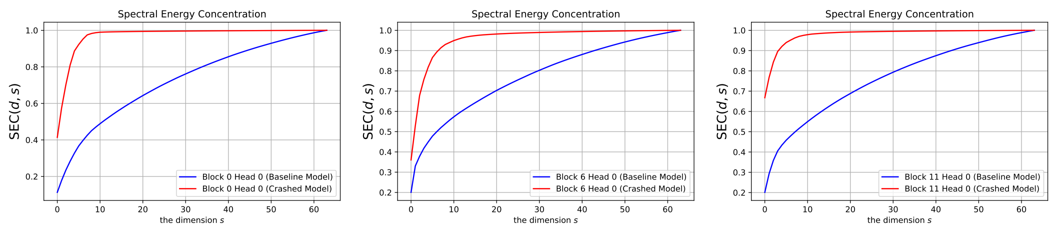
1. *Spectral energy concentration, i.e., there exists an integer s where $s \ll d_q$, such that $\mathrm{SEC}(d_q, s) \approx 1$;*

2. *A few leading singular values of $\boldsymbol{W}_q{}^\top \boldsymbol{W}_k$ are significantly large, i.e., $\sigma_i(\boldsymbol{W}_q{}^\top \boldsymbol{W}_k)$ for $i = 1, \cdots, s \ll d_q$ are very large.*

[3] For a sparse yet low-rank matrix, we refer the reader to a recent textbook (Wright & Ma, 2022).

When the malignant entropy collapse happen, the training process will crash. We provide a proof for Theorem 1 in Appendix D.

According to Theorem 1, we know that the model crash is caused by the spectral energy concentration. In Figure 4, we compares the $\text{SEC}(d_q, s)$ of three different blocks under a successfully trained model and a crashed model. In a successful training model, the spectral energy distributes in all directions. However, the spectral energy only concentrates on a few directions. As shown in Figure 4, in a crashed model the SEC collapses into less than 10 directions.



(a) *Block 0* (b) *Block 6* (c) *Block 11*

FIGURE 4: Comparison of spectral energy concentration index between a successfully trained model and a crashed model. Figure shows the results of three different blocks. The spectral energy distributes in all directions in a successful training case. The spectral energy only concentrates on a few directions in a crashed model.

$$\boldsymbol{X}^l{}_{\downarrow} \longrightarrow \boldsymbol{X}^\top \otimes \boldsymbol{X}^\top{}_{\downarrow} \longrightarrow \frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{W}_q{}^\top \boldsymbol{W}_k)}{}_{\downarrow} \longrightarrow \boldsymbol{W}_q{}^\top \boldsymbol{W}_k{}_{\downarrow} \longrightarrow \boldsymbol{A}_{\Downarrow} \longrightarrow \boldsymbol{X}^{l+1}{}_{\downarrow}$$

FIGURE 5: Attribution flow chart of attention collapse.

Figure 5 reveals how attention collapse propagates through each term, illustrating the entire attribution process from $\boldsymbol{X}^l$ as the input to the output $\boldsymbol{X}^{l+1}$ in the attention module. Note that $\downarrow$ indicates being low-rank and $\Downarrow$ means being sparse yet low-rank. If $\boldsymbol{X}$ is low-rank, then $\boldsymbol{X}^\top \otimes \boldsymbol{X}^\top$ is also low-rank because $\text{rank}(\boldsymbol{X} \otimes \boldsymbol{X}) = \text{rank}(\boldsymbol{X}) \cdot \text{rank}(\boldsymbol{X})$. According to the gradient computation, we have $\frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{W}_q^\top \boldsymbol{W}_k)} = \boldsymbol{X}^\top \otimes \boldsymbol{X}^\top$, thus we know $\frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{W}_q^\top \boldsymbol{W}_k)}$ is also low-rank. Meanwhile, it should be noted that the spectral energy of $\frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{W}_q^\top \boldsymbol{W}_k)}$ is over-concentrated, it means the gradient update will largely change $\boldsymbol{W}_q^\top \boldsymbol{W}_k$, thus $\boldsymbol{W}_q^\top \boldsymbol{W}_k$ will have large probability to be low-rank. In the paper, we have proved that being low-rank and the leading singular values of $\boldsymbol{W}_q^\top \boldsymbol{W}_k$ are very large will lead to the attention map $\boldsymbol{A}$ over-concentrated (see Appendix C for the proof.), becomes to be a sparse yet low-rank matrix. Finally, an over-concentrated $\boldsymbol{A}$ will lead to $\boldsymbol{X}^{l+1}$ to be low-rank.

3.4 OUR SOLUTION: TAMING TRANSFORMER VIA WEYL'S INQUALITY

The analysis above reveals that spectral energy concentration is the key cause leading to unstable training. One manifestation of spectral energy concentration is the rapid growth of the singular values. Therefore, our motivation is to constraint the fast growth of the singular values of the weight matrices. Fortunately, Weyl's inequality provides us a simple but effective tool.

Theorem 2 (Weyl's Inequality on Singular Values.)
Let $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathcal{R}^{m\times n}$, and $m \geq n$, and let $\sigma_1(\boldsymbol{W}_1) \geq \sigma_2(\boldsymbol{W}_1) \geq \ldots \geq \sigma_n(\boldsymbol{W}_1)$ be the ordered singular values of $\boldsymbol{W}_1$, $\sigma_i(\boldsymbol{W}_1)$ and $\sigma_i(\boldsymbol{W}_2)$ are the corresponding singular values of $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$. Then we have that:

$$\sigma_{i+j-1}(\boldsymbol{W}_1 + \boldsymbol{W}_2) \leq \sigma_i(\boldsymbol{W}_1) + \sigma_j(\boldsymbol{W}_2).$$

We provide a proof for Theorem 2 in Appendix E. From Theorem 2, it is easy to see that $\sigma_1(\boldsymbol{W}_1 + \boldsymbol{W}_2) \leq \sigma_1(\boldsymbol{W}_1) + \sigma_1(\boldsymbol{W}_2)$. Let $\boldsymbol{W}_t$ is the weight matrix at time step t, $\nabla \boldsymbol{W}_t$ is the quantity computing from gradient and its derivations (where $\nabla \boldsymbol{W}_t$ can be obtained by SGD (Robbins

Algorithm 1 AdamW^2: Taming Transformer via Weyl' Inquality without learning rate warmup.

Input: learning rate schedular α_t, weight decay λ, and first-order and second-order momentums β_1, β_2

Output: updated weight w_T

1: **for** $t = 1, 2, \ldots, T$ **do**

2: $\quad \boldsymbol{G}_t = \nabla_{\boldsymbol{W}_{t-1}} \mathcal{L}$

3: $\quad \boldsymbol{M}_t = \beta_1 \boldsymbol{M}_{t-1} + (1-\beta_1)\boldsymbol{G}_t, \quad \boldsymbol{V}_t = \beta_2 \boldsymbol{V}_{t-1} + (1-\beta_2)\boldsymbol{G}_t^2$

4: $\quad \hat{\boldsymbol{M}}_t = \boldsymbol{M}_t / (1-\beta_1^t), \quad \hat{\boldsymbol{V}}_t = \boldsymbol{V}_t / (1-\beta_2^t)$

5: $\quad \nabla \boldsymbol{W}_t = \hat{\boldsymbol{M}}_t \oslash \sqrt{\hat{\boldsymbol{V}}_t + \epsilon}$ ▷ $\nabla \boldsymbol{W}_t$ is the final update quantity.

6: $\quad \hat{\delta}_t = \mathrm{PowerIter}(\nabla \boldsymbol{W}_t), \quad \hat{\sigma}_{t-1} = \mathrm{PowerIter}(\boldsymbol{W}_{t-1})$ ▷ Power iteration to computer matrix norm.

7: $\quad$ **if** $\frac{\alpha_t \hat{\delta}_t}{\hat{\sigma}_{t-1}} > \tau$ **then** ▷ If Rule 1 is unsatisfied, readjust the learning rate.

8: $\qquad \alpha_t = \frac{\tau \hat{\sigma}_{t-1}}{\hat{\delta}_t}$

9: $\quad$ **end if**

10: $\quad$ **if** Weight Decay is Yes **then**

11: $\qquad \boldsymbol{W}_t = \boldsymbol{W}_{t-1} - \alpha_t \nabla \boldsymbol{W}_t - \alpha_t \lambda_t \boldsymbol{W}_{t-1}$

12: $\quad$ **else**

13: $\qquad \boldsymbol{W}_t = \boldsymbol{W}_{t-1} - \alpha_t \nabla \boldsymbol{W}_t$

14: $\quad$ **end if**

15: **end for**

& Monro, 1951), Adagrad (Duchi et al., 2011), or Adam (Kingma & Ba, 2014)), α_t is the learning rate at time step t. Usually, our update equation is $\boldsymbol{W}_t = \boldsymbol{W}_{t-1} - \alpha_t \nabla \boldsymbol{W}_t$. According to Weyl's Inequality, we have,

$$\sigma_1(\boldsymbol{W}_t) = \sigma_1(\boldsymbol{W}_{t-1} - \alpha_t \nabla \boldsymbol{W}_t) \leq \sigma_1(\boldsymbol{W}_{t-1}) + \alpha_t \sigma_1(\nabla \boldsymbol{W}_t). \tag{5}$$

A key problem in Equation 5 is that if $\sigma_1(\nabla \boldsymbol{W})$ is very large, then $\boldsymbol{W}_t$ will be largely different from $\boldsymbol{W}_{t-1}$. It means this update at time step t will not be smooth. A smooth update should satisfy the following rule.

Rule 1 (Smooth Weight Updating Rule)
Given $\boldsymbol{W}_{t-1}$ and the updating quantity at step t, with the learning rate α_t, a smooth update should satisfy the following inequality: $\|\boldsymbol{W}_{t-1} - \alpha_t \nabla \boldsymbol{W}_t\|_2 \leq (1+\tau)\|\boldsymbol{W}_{t-1}\|_2$, where τ is a small factor.

If the Rule 1 is satisfied, then it means that $\sigma_1(\boldsymbol{W}_{t-1}) + \alpha_t \sigma_1(\nabla \boldsymbol{W}_t) \leq (1+\tau)\|\boldsymbol{W}_{t-1}\|_2 = (1+\tau)\sigma_1(\boldsymbol{W}_{t-1})$. We can derive the following inequality,

$$\alpha_t \leq \tau \frac{\sigma_1(\boldsymbol{W}_{t-1})}{\sigma_1(\nabla \boldsymbol{W}_t)}. \tag{6}$$

This inequality depicts that the learning rate should be bounded by a ratio of singular values of $\sigma_1(\boldsymbol{W}_{t-1})$, $\sigma_1(\nabla \boldsymbol{W}_t)$. Generally, τ is a small value, *e.g.*, 0.004 or 0.005. The intuition behind this is that if the spectral norm of $\nabla \boldsymbol{W}$ are significantly larger than that of $\boldsymbol{W}$, then the model is potentially undergoing rapid changes. In such cases, a large learning rate could lead to training instability. Therefore, our motivation is that if $\alpha_t \frac{\sigma_1(\nabla \boldsymbol{W}_t)}{\sigma_1(\boldsymbol{W}_{t-1})} > \tau$, then α_t will be truncated to $\tau \frac{\sigma_1(\boldsymbol{W}_{t-1})}{\sigma_1(\nabla \boldsymbol{W}_t)}$. Since our based optimizer is AdamW (Loshchilov & Hutter, 2019) and our algorithm is motivated by Weyl's Inequality, thus, we term our algorithm as AdamW^2.

Algorithm 1 shows our AdamW^2 optimizer. Lines 6-9 marks our improvement to the base optimizer, the other codes are same as AdamW. According to line 6 in Algorithm 1, $\sigma_1(\nabla \boldsymbol{W}_t)$ and $\sigma_1(\boldsymbol{W}_{t-1})$ is computed via a fast power iteration method. In practice, we set the maximum iterations in power iteration to 3. Actually, we find two iterations are enough to estimate the spectral norm of matrices. According to Equation 6, if $\alpha_t > \tau \frac{\sigma_1(\boldsymbol{W}_{t-1})}{\sigma_1(\nabla \boldsymbol{W}_t)}$, then the learning rate α_t will be truncated to $\tau \frac{\sigma_1(\boldsymbol{W}_{t-1})}{\sigma_1(\nabla \boldsymbol{W}_t)}$, or the algorithm will adjust α_t and use the default learning rate set by the learning rate schedule. Our core operation corresponds to lines 7-8 in Algorithm 1.

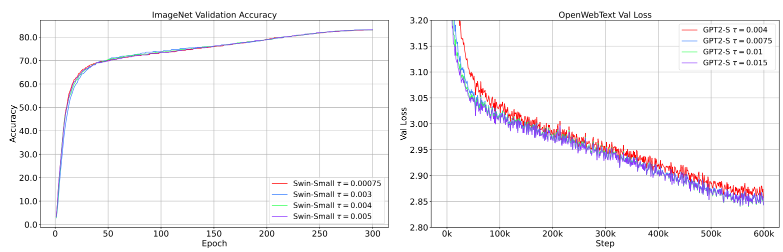
4 Experiments

Compared to some previous works (Bachlechner et al., 2021; Wang et al., 2019; Xiong et al., 2020; Wang et al., 2022; Qi et al., 2023b) that focuses on improving the training stability of Transformer,

TABLE 1: Quantitative comparison of AdamW and AdamW^2 with and without learning rate warmup. AdamW^2 demonstrates a very competitive performance compared to AdamW.

Method	**ViT (Acc. ↑)**		**GPT (Loss ↓)**	**Swin-Transformer (Acc. ↑)**	
Configurations	ViT-B	ViT-L	GPT-S	Swin-S	Swin-B
Parameters	86M	307M	125M	50M	88M
AdamW (with warmup)	80.22	81.65	2.848	83.02	83.48
AdamW^2 (no warmup)	80.58	81.82	2.840	83.14	83.44

our method do not need to adjust the network structure and we do not use learning rate warmup. For ViT, Swin-Transformer and GPT, we will use a warmup of 60 epochs, 20 epochs and 2000 steps individually. In AdamW^2, we directly use a cosine learning rate schedule and decay the learning rate from maximum to minimum.



(a) Swin-Transformer

(b) GPT

FIGURE 6: Ablation study of τ in AdamW^2 using Swin-S and GPT-S.

We conduct experiments on three popular Transformers, *i.e.*, ViT (Dosovitskiy et al., 2020), GPT-2 (Radford et al., 2019) and Swin-Transformer (Liu et al., 2021), where ViT are Swin-Transformer are pure encoder architectures and GPT is a pure casual decoder. Note that we do not conduct any adjustment to the networks and directly use the original implementation. Our experiments include image classification on ImageNet (Deng et al., 2009) and large language model on OpenWebText (Gokaslan & Cohen) dataset. We list some training configuration in Appendix N. The quantitative results are shown in Table 1. Our baseline model is the corresponding Transformer using a learning rate warmup; whereas baseline models without using learning rate warmup will crash. AdamW^2 demonstrates a very competitive performance compared to the baseline method. These experimental results further verify that our previous understanding to the training dynamics of Transformer is rational.

We also conduct ablation study of the choice of τ in GPT and Swin-Transformer. The results are shown in Figure 6. We can see that the performance of AdamW^2 varies slightly for different values of τ, but overall, our approach is robust for different choices of τ. The lines basically overlap in the later period because our smooth updating rule is never broken in the later period.

5 CONCLUSION

In this paper, we revisited the training dynamics of Transformers by visualizing the spectral norm of weight matrices, the activations and the attention map, presented a theoretical analysis for Transformer training and identified two attention entropy collapse modes, *i.e.*, the benign collapse and the malignant collapse, in which the malignant collapse accompanies with model crash. Moreover, we revealed that the *spectral energy concentration* of ${\boldsymbol{W}_q}^\top \boldsymbol{W}_k$ is the key problem behind the model crash, which causes the attention map to be sparse yet low-rank. Furthermore, we proposed a smoothing rule to resolve the problem of spectral energy concentration of ${\boldsymbol{W}_q}^\top \boldsymbol{W}_k$ by controlling the rapid growth of singular values, which can prevent the fast spectral energy concentration to a few directions and thus avoid the malignant entropy collapse. We conducted extensive experiments to verify the proposed strategy with ViT, Swin Transformer and GPT and demonstrated that the proposed strategy could effectively and stably train a model without using any learning rate warmup.

Ethics Statement

In this paper, we aim to provide a novel approach to train transformer without learning rate warmup. Our work does not involve any human subjects, and we have carefully ensured that it poses no potential risks or harms. Additionally, there are no conflicts of interest, sponsorship concerns, or issues related to discrimination, bias, or fairness associated with this study. We have taken steps to address privacy and security concerns, and all data used comply with legal and ethical standards. Our work fully adheres to research integrity principles, and no ethical concerns have arisen during the course of this study.

Reproducibility Statement

To ensure the reproducibility of our work, we provide all the details to reproduce the experiments. Theoretical proofs of the claims made in this paper, and detailed experimental settings and configurations are provided in Appendices.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*, pp. 1352–1361. PMLR, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113, 2023.

Aaron Defazio, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, Ashok Cutkosky, et al. The road less scheduled. *arXiv preprint arXiv:2405.15682*, 2024.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pp. 7480–7512. PMLR, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, pp. 2793–2803. PMLR, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus.

Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

Alexander Graham. *Kronecker products and matrix calculus with applications*. Courier Dover Publications, 2018.

Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. Flatten transformer: Vision transformer using focused linear attention. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5961–5971, 2023.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4246–4253, 2020.

Roger A Horn and Charles R Johnson. Topics in matrix analysis, 1991. *Cambridge University Presss, Cambridge*, 37:39, 1991.

Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

Andrej Karpathy. NanoGPT. `https://github.com/karpathy/nanoGPT`, 2022.

Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. In *International Conference on Machine Learning*, pp. 5562–5571. PMLR, 2021.

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

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–50. Springer, 2002.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5747–5763, 2020.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. In *International Conference on Learning Representations*, 2019.

Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. *Advances in Neural Information Processing Systems*, 35:27198–27211, 2022.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

Xianbiao Qi, Jianan Wang, Yihao Chen, Yukai Shi, and Lei Zhang. Lipsformer: Introducing lipschitz continuity to vision transformers. In *The Eleventh International Conference on Learning Representations*, 2023a.

Xianbiao Qi, Jianan Wang, and Lei Zhang. Understanding optimization of deep learning via jacobian matrix and lipschitz constant. *arXiv preprint arXiv:2306.09338*, 2023b.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pp. 8821–8831. Pmlr, 2021.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pp. 464–472. IEEE, 2017.

Yuandong Tian, Yiping Wang, Beidi Chen, and Simon S Du. Scan and snap: Understanding training dynamics and token composition in 1-layer transformer. *Advances in Neural Information Processing Systems*, 36:71911–71947, 2023a.

Yuandong Tian, Yiping Wang, Zhenyu Zhang, Beidi Chen, and Simon Du. Joma: Demystifying multilayer transformers via joint dynamics of mlp and attention. *arXiv preprint arXiv:2310.00535*, 2023b.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *arXiv preprint arXiv:2203.00555*, 2022.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787*, 2019.

Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

John Wright and Yi Ma. *High-dimensional data analysis with low-dimensional models: Principles, computation, and applications.* Cambridge University Press, 2022.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020.

Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Joshua M Susskind. Stabilizing transformer training by preventing attention entropy collapse. In *International Conference on Machine Learning*, pp. 40770–40803. PMLR, 2023.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

A Kronecker Product and Vectorization

Kronecker product (Graham, 2018; Petersen et al., 2008), also called as matrix direct product, is an operation defined on two matrices of arbitrary size. The specific definition is as follows.

Definition 1 (Kronecker Product)
Let $\boldsymbol{A}$ be an $n \times p$ matrix and $\boldsymbol{B}$ an $m \times q$ matrix. The $mn \times pq$ matrix

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{1,1}\boldsymbol{B} & a_{1,2}\boldsymbol{B} & \cdots & a_{1,p}\boldsymbol{B} \\ a_{2,1}\boldsymbol{B} & a_{2,2}\boldsymbol{B} & \cdots & a_{2,p}\boldsymbol{B} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1}\boldsymbol{B} & a_{n,2}\boldsymbol{B} & \cdots & a_{n,p}\boldsymbol{B} \end{bmatrix}$$

is called the Kronecker product of $\boldsymbol{A}$ and $\boldsymbol{B}$. It is also called the direct product or the tensor product.

For instance, if $\boldsymbol{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, and $\boldsymbol{B} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \end{bmatrix}$, then $\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} 1 & 2 & 3 & 2 & 4 & 6 \\ 3 & 4 & 5 & 6 & 8 & 10 \\ 3 & 6 & 9 & 4 & 8 & 12 \\ 9 & 12 & 15 & 12 & 16 & 20 \end{bmatrix}$.

Some basic properties of Kronecker product includes

$$\boldsymbol{A} \otimes (\boldsymbol{B} \otimes \boldsymbol{C}) = (\boldsymbol{A} \otimes \boldsymbol{B}) \otimes \boldsymbol{C},$$
$$\boldsymbol{A} \otimes (\boldsymbol{B} + \boldsymbol{C}) = (\boldsymbol{A} \otimes \boldsymbol{B}) + (\boldsymbol{A} \otimes \boldsymbol{C}), \quad (\boldsymbol{A} + \boldsymbol{B}) \otimes \boldsymbol{C} = (\boldsymbol{A} \otimes \boldsymbol{C}) + (\boldsymbol{B} \otimes \boldsymbol{C}),$$
$$(\boldsymbol{A} \otimes \boldsymbol{B})^T = \boldsymbol{A}^T \otimes \boldsymbol{B}^T.$$

For a matrix $\boldsymbol{A}$, the rank of $\boldsymbol{A} \otimes \boldsymbol{A}$ can be computed as,

$$\operatorname{rank}(\boldsymbol{A} \otimes \boldsymbol{A}) = \operatorname{rank}(\boldsymbol{A}) \cdot \operatorname{rank}(\boldsymbol{A}).$$

It means if the rank of the matrix $\boldsymbol{A}$ is small, then the rank of $\boldsymbol{A} \otimes \boldsymbol{A}$ will also be very small.

In mathematics, *Vectorization* (Graham, 2018; Petersen et al., 2008) is usually used together with the Kronecker product to express matrix multiplication as a linear transformation on matrices. After vectorization, we can calculate Jacobian matrix of matrix product more conveniently. A property of vectorization for matrix product is defined below.

Proposition 2 (Property of Vectorization for Matrix Product)
Let $\boldsymbol{A} \in \mathcal{R}^{m\times n}, \boldsymbol{B} \in \mathcal{R}^{n\times k}, \boldsymbol{C} \in \mathcal{R}^{k\times l}$, then we have

$$\operatorname{vec}(\boldsymbol{ABC}) = (\boldsymbol{C}^\top \otimes \boldsymbol{A}) \operatorname{vec}(\boldsymbol{B}).$$

Proof. Let $\boldsymbol{C}_i$ be the i-th row of $\boldsymbol{C}$. Then we have:

$$\begin{aligned} \operatorname{vec}(\boldsymbol{ABC}) &= \sum_{i=1}^{n} \sum_{j=1}^{k} b_{ij} \operatorname{vec}(\boldsymbol{a}_i \boldsymbol{C}_j) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{k} b_{ij} (\boldsymbol{C}_j^\top \otimes \boldsymbol{a}_i) \\ &= \sum_{j=1}^{k} (\boldsymbol{C}_j^\top \otimes \boldsymbol{A}) \boldsymbol{b}_j \\ &= (\boldsymbol{C}^\top \otimes \boldsymbol{A}) \operatorname{vec}(\boldsymbol{B}). \end{aligned}$$

□

Furthermore, we have the following properties:

$$\operatorname{vec}(\boldsymbol{AB}) = (\boldsymbol{I}_k \otimes \boldsymbol{A}) \operatorname{vec}(\boldsymbol{B}) = \left(\boldsymbol{B}^\top \otimes \boldsymbol{I}_m\right) \operatorname{vec}(\boldsymbol{A}),$$
$$\begin{aligned} \operatorname{vec}(\boldsymbol{ABC}) &= \left(\boldsymbol{C}^\top \boldsymbol{B}^\top \otimes \boldsymbol{I}_m\right) \operatorname{vec}(\boldsymbol{A}) \\ &= \left(\boldsymbol{C}^\top \otimes \boldsymbol{A}\right) \operatorname{vec}(\boldsymbol{B}) \\ &= (\boldsymbol{I}_l \otimes \boldsymbol{AB}) \operatorname{vec}(\boldsymbol{C}) \end{aligned}$$

where $\boldsymbol{I}_k \in \mathcal{R}^{k\times k}, \boldsymbol{I}_l \in \mathcal{R}^{l\times l}, \boldsymbol{I}_m \in \mathcal{R}^{m\times m}$ are all identity matrices.

Together with Kronecker product, vectorization is an effective tool to compute matrix calculus. We can see the following two examples.

Let $\boldsymbol{P} = \boldsymbol{A}\boldsymbol{B}$ where $\boldsymbol{A} \in \mathcal{R}^{m\times n}, \boldsymbol{B} \in \mathcal{R}^{n\times k}$, we have

$$\frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{A})} = \boldsymbol{B}^\top \otimes \boldsymbol{I}_m, \quad \frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{B})} = \boldsymbol{I}_k \otimes \boldsymbol{A}.$$

Let $\boldsymbol{P} = \boldsymbol{A}\boldsymbol{B}\boldsymbol{C}$ where $\boldsymbol{A} \in \mathcal{R}^{m\times n}, \boldsymbol{B} \in \mathcal{R}^{n\times k}, \boldsymbol{C} \in \mathcal{R}^{k\times l}$, we have,

$$\frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{A})} = \boldsymbol{C}^\top \boldsymbol{B}^\top \otimes \boldsymbol{I}_m, \quad \frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{B})} = \boldsymbol{C}^\top \otimes \boldsymbol{A}, \quad \frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{C})} = \boldsymbol{I}_l \otimes \boldsymbol{A}\boldsymbol{B}.$$

Vectorization and Kronecker product provide us a convenient way to analyze the self-attention module. We can compute the Jacobian matrix of the output with respect to the input or the weight matrix more conveniently. For more introduction to Kronecker product and vectorization, the readers can refer to (Petersen et al., 2008; Graham, 2018)

B DERIVATION OF JACOBIAN MATRIX FOR SINGLE-HEAD SELF-ATTENTION

A single-head self-attention can be defined as

$$\boldsymbol{Y} = \boldsymbol{W}_v \boldsymbol{X} \boldsymbol{A},$$

where $\boldsymbol{P} = \boldsymbol{X}^\top {\boldsymbol{W}_q}^\top \boldsymbol{W}_k \boldsymbol{X}, \quad \boldsymbol{A} = \text{softmax}(\frac{\boldsymbol{P}}{\sqrt{d_q}})$. $\boldsymbol{A}$ is called as the attention matrix and $\frac{\boldsymbol{P}}{\sqrt{d_q}}$ is called as the logit, $\boldsymbol{A} \in \mathcal{R}^{n\times n}, \boldsymbol{X} \in \mathcal{R}^{d\times n}, \boldsymbol{W}_v \in \mathcal{R}^{d_v\times d}$. Here, our goal is to calculate $\frac{\partial \text{vec}(\boldsymbol{Y})}{\partial \text{vec}(\boldsymbol{X})}$.

In the main body, we have derived $\frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{X})}$. Here, let us calculate the matrix calculus of $\boldsymbol{A} = \text{softmax}(\frac{\boldsymbol{P}}{\sqrt{d_q}})$ with respect to $\boldsymbol{P}$ using Kronecker products and vectorization. We can rewrite it as $\boldsymbol{A} = \exp(\frac{\boldsymbol{P}}{\sqrt{d_q}}) \oslash (\boldsymbol{1}_n \boldsymbol{1}_n^\top \exp(\frac{\boldsymbol{P}}{\sqrt{d_q}}))$, where $\boldsymbol{1}_n$ denotes a $\mathcal{R}^n$ all one vector. Note that $\boldsymbol{A}$ is obtained by conduct a softmax operation in each column individually.

First, let us define two intermediate variables:

$$\boldsymbol{B} = \exp(\frac{\boldsymbol{P}}{\sqrt{d_q}}), \quad \boldsymbol{C} = \boldsymbol{1}\boldsymbol{1}^\top \exp(\frac{\boldsymbol{P}}{\sqrt{d_q}}) = \boldsymbol{1}\boldsymbol{1}^\top \boldsymbol{B}.$$

In this way, we can represent the attention matrix $\boldsymbol{A}$ as $\boldsymbol{A} = \boldsymbol{B} \oslash \boldsymbol{C}$.

Then, we can vectorize the equation,

$$\text{vec}(\boldsymbol{A}) = \text{vec}(\boldsymbol{B} \oslash \boldsymbol{C}) = \text{vec}(\boldsymbol{B}) \oslash \text{vec}(\boldsymbol{C}).$$

According to the chain rule, we have

$$\frac{\partial \text{vec}(\boldsymbol{A})}{\partial \text{vec}(\boldsymbol{P})} = \frac{\partial \text{vec}(\boldsymbol{A})}{\partial \text{vec}(\boldsymbol{B})} \frac{\partial \text{vec}(\boldsymbol{B})}{\partial \text{vec}(\boldsymbol{P})} + \frac{\partial \text{vec}(\boldsymbol{A})}{\partial \text{vec}(\boldsymbol{C})} \frac{\partial \text{vec}(\boldsymbol{C})}{\partial \text{vec}(\boldsymbol{P})}.$$

Let us calculate each individual term. We have

$$\frac{\partial \text{vec}(\boldsymbol{A})}{\partial \text{vec}(\boldsymbol{B})} = \mathbf{1}_{n^2} \oslash \text{diag}(\text{vec}(\boldsymbol{C})),$$
$$\frac{\partial \text{vec}(\boldsymbol{B})}{\partial \text{vec}(\boldsymbol{P})} = \frac{\text{diag}(\text{vec}(\boldsymbol{B}))}{\sqrt{d_q}},$$
$$\frac{\partial \text{vec}(\boldsymbol{A})}{\partial \text{vec}(\boldsymbol{C})} = -\text{diag}(\text{vec}(\boldsymbol{B}) \oslash (\text{vec}(\boldsymbol{C}) \odot \text{vec}(\boldsymbol{C}))),$$
$$\frac{\partial \text{vec}(\boldsymbol{C})}{\partial \text{vec}(\boldsymbol{P})} = \frac{(\boldsymbol{I}_n \otimes \mathbf{1}_n \mathbf{1}_n^\top)\text{diag}(\text{vec}(\boldsymbol{B}))}{\sqrt{d_q}},$$

where $\boldsymbol{I}_n$ is a $n \times n$ identity matrix and $\otimes$ is the Kronecker product, $\mathbf{1}_{nn}$ denotes a $\mathcal{R}^{n^2}$ all one vector.

Substitute these four term into the chain rule, we have

$$\begin{aligned}\frac{\partial \text{vec}(\boldsymbol{A})}{\partial \text{vec}(\boldsymbol{P})} &= \frac{\left(\mathbf{1}_{n^2} \oslash \text{diag}(\text{vec}(\boldsymbol{C}))\right)\text{diag}(\text{vec}(\boldsymbol{B})) - \text{diag}(\text{vec}(\boldsymbol{B}) \oslash (\text{vec}(\boldsymbol{C}) \odot \text{vec}(\boldsymbol{C})))(\boldsymbol{I}_n \otimes \mathbf{1}_n \mathbf{1}_n^\top)\text{diag}(\text{vec}(\boldsymbol{B}))}{\sqrt{d_q}} \\ &= \frac{\text{diag}(\text{vec}(\boldsymbol{A})) - \text{diag}(\text{vec}(\boldsymbol{B}) \oslash (\text{vec}(\boldsymbol{C}) \odot \text{vec}(\boldsymbol{C})))(\boldsymbol{I}_n \otimes \mathbf{1}_n \mathbf{1}_n^\top)\text{diag}(\text{vec}(\boldsymbol{B}))}{\sqrt{d_q}} \\ &= \frac{\text{diag}(\text{vec}(\boldsymbol{A})) - (\boldsymbol{A} \otimes \boldsymbol{I}_n)\text{diag}(\text{vec}(\boldsymbol{A}))}{\sqrt{d_q}}.\end{aligned}$$

If $\boldsymbol{A}$ and $\boldsymbol{P}$ are two vectors, here, let us use $\boldsymbol{a}$ and $\boldsymbol{p}$ denote them individually, then we know

$$\frac{\partial \text{vec}(\boldsymbol{a})}{\partial \text{vec}(\boldsymbol{p})} = \frac{\text{diag}(\boldsymbol{a}) - \boldsymbol{a}\boldsymbol{a}^\top}{\sqrt{d_q}}.$$

If $\boldsymbol{a}$ approaches to a unit vector $\boldsymbol{e}$, then the Jabobian matrix $\frac{\partial \text{vec}(\boldsymbol{a})}{\partial \text{vec}(\boldsymbol{p})}$ will tend to $\mathbf{0}$.

In Section 3.2, we have the following Jacobian matrix

$$\frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{X})} = (\boldsymbol{X}^\top \boldsymbol{W}_q{}^\top \boldsymbol{W}_k \otimes \boldsymbol{I}_n)\boldsymbol{K} + (\boldsymbol{I}_n \otimes \boldsymbol{X}^\top \boldsymbol{W}_q{}^\top \boldsymbol{W}_k).$$

By vectorization of $\boldsymbol{Y} = \boldsymbol{W}_v \boldsymbol{X} \boldsymbol{A}$, we have

$$\text{vec}(\boldsymbol{Y}) = (\boldsymbol{A}^\top \otimes \boldsymbol{W}_v)\text{vec}(\boldsymbol{X}) + (\boldsymbol{I}_n \otimes \boldsymbol{W}_v \boldsymbol{X})\text{vec}(\boldsymbol{A}).$$

Therefore, according to the product rule and chain rule, we can denote the Jacobian matrix of $\boldsymbol{Y}$ with respect to $\boldsymbol{X}$ as follows:

$$\begin{aligned}\frac{\partial \text{vec}(\boldsymbol{Y})}{\partial \text{vec}(\boldsymbol{X})} &= (\boldsymbol{A}^\top \otimes \boldsymbol{W}_v) + (\boldsymbol{I}_n \otimes \boldsymbol{W}_v \boldsymbol{X})\frac{\partial \text{vec}(\boldsymbol{A})}{\partial \text{vec}(\boldsymbol{X})}, \\ &= (\boldsymbol{A}^\top \otimes \boldsymbol{W}_v) + (\boldsymbol{I}_n \otimes \boldsymbol{W}_v \boldsymbol{X})\frac{\partial \text{vec}(\boldsymbol{A})}{\partial \text{vec}(\boldsymbol{P})}\frac{\partial \text{vec}(\boldsymbol{P})}{\partial \text{vec}(\boldsymbol{X})}.\end{aligned}$$

Bringing in all the terms, we get the following formula

$$\frac{\partial \text{vec}(\boldsymbol{Y})}{\partial \text{vec}(\boldsymbol{X})} = (\boldsymbol{A}^\top \otimes \boldsymbol{W}_v) + (\boldsymbol{I}_n \otimes \boldsymbol{W}_v \boldsymbol{X})\left(\frac{\text{diag}(\text{vec}(\boldsymbol{A})) - (\boldsymbol{A} \otimes \boldsymbol{I}_n)\text{diag}(\text{vec}(\boldsymbol{A}))}{\sqrt{d_q}}\right)\left((\boldsymbol{X}^\top \boldsymbol{W}_q{}^\top \boldsymbol{W}_k \otimes \boldsymbol{I}_n)\boldsymbol{K} + (\boldsymbol{I}_n \otimes \boldsymbol{X}^\top \boldsymbol{W}_q{}^\top \boldsymbol{W}_k)\right). \tag{7}$$

Let us analyze Equation 7. If a malignant entropy mode happens, $\frac{\text{diag}(\text{vec}(\boldsymbol{A}))-(\boldsymbol{A}\otimes\boldsymbol{I}_n)\text{diag}(\text{vec}(\boldsymbol{A}))}{\sqrt{d_q}}$ will approach to $\mathbf{0}$ because each $\boldsymbol{a}$ in $\boldsymbol{A}$ will be a unit vector $\boldsymbol{e}$. From the perspective of forward process, the features $\boldsymbol{Y}$ will collapse to several directions. From the perspective of backward process, $\frac{\text{diag}(\text{vec}(\boldsymbol{A}))-(\boldsymbol{A}\otimes\boldsymbol{I}_n)\text{diag}(\text{vec}(\boldsymbol{A}))}{\sqrt{d_q}}$ will become $\mathbf{0}$, and $\frac{\partial\text{vec}(\boldsymbol{Y})}{\partial\text{vec}(\boldsymbol{X})}$ will be a sparse low-rank matrix. Through $\boldsymbol{A}^\top\otimes\boldsymbol{W}_v$, most of position in $\boldsymbol{X}$ will get zero gradient, and only very few columns will obtain some large noisy gradients. In a malignant entropy mode, the learned feature is invalid and useless. Similarly, if a benign entropy mode happens, the attention map $\boldsymbol{A}$ will approach to an identity matrix $\boldsymbol{I}$ and $\frac{\text{diag}(\text{vec}(\boldsymbol{A}))-(\boldsymbol{A}\otimes\boldsymbol{I}_n)\text{diag}(\text{vec}(\boldsymbol{A}))}{\sqrt{d_q}}\approx 0$ when $\boldsymbol{A}\approx\boldsymbol{I}$. Therefore, we have $\frac{\partial\text{vec}(\boldsymbol{Y})}{\partial\text{vec}(\boldsymbol{X})}\approx(\boldsymbol{I}^\top\otimes\boldsymbol{W}_v)$. In this way, a self-attention module degenerates to a linear layer.

C Proof of Benign Entropy Collapse

Recall that $\boldsymbol{P}=\boldsymbol{X}^\top\boldsymbol{W}_q{}^\top\boldsymbol{W}_k\boldsymbol{X}, \boldsymbol{A}=\text{softmax}(\frac{\boldsymbol{P}}{\sqrt{d_q}})$. Here, let $\boldsymbol{W}=\boldsymbol{W}_q{}^\top\boldsymbol{W}_k$ and $\boldsymbol{W}\in\mathcal{R}^{d\times d}$. In this way, we denote that $\boldsymbol{A}=\text{softmax}(\frac{\boldsymbol{X}^\top\boldsymbol{W}\boldsymbol{X}}{\sqrt{d_q}})$. We know that $\text{rank}(\boldsymbol{W})\le d_q$ and $d_q<d$. To prove $\boldsymbol{A}$ will always collapse to an identity matrix when $\boldsymbol{W}$ is a non-symmetric positive quasi-definite square matrix, it is equivalent to prove $\mathbb{E}\left[\boldsymbol{x}_i{}^\top\boldsymbol{W}\boldsymbol{x}_i\right]\gg\mathbb{E}\left[\boldsymbol{x}_i{}^\top\boldsymbol{W}\boldsymbol{x}_j\right]$ for any $i\neq j$. It will be very hard to prove it mathematically if $\boldsymbol{W}$ is a form of a non-symmetric positive quasi-definite square matrix. Therefore, let us make some simplification assumptions. Assume $\boldsymbol{W}$ is a real symmetric positive semi-definite square matrix and its trace is in direct proportion to the dimension d_q, and any $\boldsymbol{x}_i$ is a high-dimension random vector and each element in $x_{i,j}\overset{\text{iid}}{\sim}\mathcal{N}(0,1)$.

We break our proof into a sub-problems.

Proposition 3 (Expectation of $\boldsymbol{x}_i{}^\top\boldsymbol{W}\boldsymbol{x}_i$)
Let $\boldsymbol{W}$ is a real symmetric positive semi-definite matrix, and any $\boldsymbol{x}_i$ is a high-dimension random vector, $\mathbb{E}\left[\boldsymbol{x}_i{}^\top\boldsymbol{W}\boldsymbol{x}_i\right]=\text{trace}(\boldsymbol{W})$.

Proof. Let $\boldsymbol{W}$ be a real symmetric positive semi-definite, thus it can be decomposed into $\boldsymbol{W}=\boldsymbol{U}\Sigma\boldsymbol{U}^\top$. In this way, we have

$$\begin{aligned}
\mathbb{E}\left[\boldsymbol{x}_i{}^\top\boldsymbol{W}\boldsymbol{x}_i\right] &= \mathbb{E}\left[\boldsymbol{x}_i{}^\top\boldsymbol{U}\Sigma\boldsymbol{U}^\top\boldsymbol{x}_i\right]\\
&= \mathbb{E}\left[\boldsymbol{z}^\top\Sigma\boldsymbol{z}\quad(\text{let } \boldsymbol{z}=\boldsymbol{U}^\top\boldsymbol{x}_i)\right]\\
&= \mathbb{E}\left[\sum_{i=1}^{d}\sigma_i\boldsymbol{z}^2\right]\quad(\Sigma\text{ is a diagonal matrix})\\
&= \sum_{i=1}^{d_q}\sigma_i\times(0+1)\quad(\text{by independence, mean 0})\\
&= \sum_{i=1}^{d_q}\sigma_i=\text{trace}(\boldsymbol{W}).
\end{aligned}$$

For a real symmetric positive semi-definite, all its singular values is larger or equal to 0. Thus, we know $\sum_{i=1}^{d_q}\sigma_i>0$ considering $\boldsymbol{W}$ is not a all zero matrix. □

Proposition 4 (Expectation of $\boldsymbol{x}_i{}^\top\boldsymbol{W}\boldsymbol{x}_j$ for $i\neq j$)
Let $\boldsymbol{W}$ is a real symmetric positive semi-definite square matrix, and any $\boldsymbol{x}_i$ is a high-dimension random vector, $\mathbb{E}_{i\neq j}\left[\boldsymbol{x}_i{}^\top\boldsymbol{W}\boldsymbol{x}_j\right]=0$.

Proof. $\boldsymbol{W}$ is a real symmetric positive semi-definite, thus it can be decomposed into $\boldsymbol{W} = \boldsymbol{U}\Sigma\boldsymbol{U}^\top$. In this way, we have

$$
\begin{aligned}
\mathbb{E}_{i\neq j}\left[\boldsymbol{x}_i{}^\top \boldsymbol{W}\boldsymbol{x}_j\right] &= \mathbb{E}_{i\neq j}\left[\boldsymbol{x}_i{}^\top \boldsymbol{U}\Sigma\boldsymbol{U}^\top\boldsymbol{x}_j\right] \\
&= \mathbb{E}\left[\boldsymbol{z}^\top\Sigma\boldsymbol{v}\right] \quad (\text{let } \boldsymbol{z} = \boldsymbol{U}^\top\boldsymbol{x}_i, \text{ and } \boldsymbol{v} = \boldsymbol{U}^\top\boldsymbol{x}_j) \\
&= \mathbb{E}\left[\sum_{i=1}^{d_q}\sigma_i\boldsymbol{z}^\top\boldsymbol{v}\right] \quad (\Sigma \text{ is a diagonal matrix. } \boldsymbol{z} \text{ and } \boldsymbol{v} \text{ are independent}) \\
&= 0. \quad (\text{According to vecotr concentration inequality } \boldsymbol{z}^\top\boldsymbol{v} \to 0 \text{ for high dimension } d)
\end{aligned}
$$

□

According to Proposition 3 and Proposition 4, we can have that $\mathbb{E}\left[\boldsymbol{x}_i{}^\top\boldsymbol{W}\boldsymbol{x}_i\right] > \mathbb{E}\left[\boldsymbol{x}_i{}^\top\boldsymbol{W}\boldsymbol{x}_j\right]$ for any $i \neq j$. Considering that $\boldsymbol{W}$ is usually a high-dimensional matrix and some of its singular values are significantly larger than 0, thus, after softmax operation, $\boldsymbol{A}$ will always collapse to an identity matrix. In this way, the self-attention module degenerates into a linear projection module. The model fitting ability will decline, but model training will not crash. Our proof is based on matrix computations (Golub & Van Loan, 2013) and high-dimensional probability (Vershynin, 2018), readers can refer to these two books.

D Proof of Malignant Entropy Collapse

Proof. Recall that we have $\boldsymbol{P} = \boldsymbol{X}^\top\boldsymbol{W}_q{}^\top\boldsymbol{W}_k\boldsymbol{X}, \quad \boldsymbol{A} = \text{softmax}(\frac{\boldsymbol{P}}{\sqrt{d_q}})$. Let $\boldsymbol{W} = \boldsymbol{W}_q{}^\top\boldsymbol{W}_k$ and $\boldsymbol{W} \in \mathcal{R}^{d\times d}$. Let us take a column of $\boldsymbol{P}$, we have $\boldsymbol{P}_{:,i} = \boldsymbol{X}^\top\boldsymbol{W}\boldsymbol{x}_i$. According to SVD decomposition, we have $\boldsymbol{W} = \boldsymbol{U}\Sigma\boldsymbol{V}^\top$. Since in the malignant mode of entropy collapse, the matrix $\boldsymbol{W}$ has a significantly lower rank than d. Here, let us assume rank$(\boldsymbol{W}) = k$, and $k \ll d$. We know that $\sigma_j(\boldsymbol{W}) = 0$, where $j > k$. For convenience, let us just write σ_j for $\sigma_j(\boldsymbol{W})$. Thus, we have

$$
\begin{aligned}
\boldsymbol{P}_{:,i} &= \boldsymbol{X}^\top\boldsymbol{W}\boldsymbol{x}_i \\
&= \boldsymbol{X}^\top\boldsymbol{U}\Sigma\boldsymbol{V}^\top\boldsymbol{x}_i \\
&= [\sigma_1\boldsymbol{z}_1{}^\top\boldsymbol{q}_i, \quad \sigma_2\boldsymbol{z}_2{}^\top\boldsymbol{q}_i, \quad ..., \quad \sigma_d\boldsymbol{z}_d{}^\top\boldsymbol{q}_i]^\top \quad (\text{let } \boldsymbol{z}_j = \boldsymbol{U}^\top\boldsymbol{x}_j \text{ and } \boldsymbol{q}_i = \boldsymbol{V}^\top\boldsymbol{x}_i) \\
&= [\sigma_1\boldsymbol{z}_1{}^\top\boldsymbol{q}_i, \quad ..., \quad \sigma_k\boldsymbol{z}_k{}^\top\boldsymbol{q}_i, \quad 0\times\boldsymbol{z}_{k+1}{}^\top\boldsymbol{q}_i, \quad ..., \quad ,0\times\boldsymbol{z}_d{}^\top\boldsymbol{q}_i]^\top.
\end{aligned}
$$

Let us assume that after normalization (*e.g.*, LayerNorm (Ba et al., 2016), RMSNorm (Zhang & Sennrich, 2019)), each element in $z_{j,l} \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$ and $q_{i,l} \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$. We know $\boldsymbol{z}, \boldsymbol{q} \in \mathcal{R}^d$, we can easily derive that $\boldsymbol{z}^\top\boldsymbol{q} \overset{\text{iid}}{\sim} \mathcal{N}(0,d)$. In this way, we have

$$
\begin{bmatrix}
\sigma_1\boldsymbol{z}_1{}^\top\boldsymbol{q}_i \\
\cdots \\
\sigma_k\boldsymbol{z}_k{}^\top\boldsymbol{q}_i \\
0\times\boldsymbol{z}_{k+1}{}^\top\boldsymbol{q}_i \\
\cdots \\
0\times\boldsymbol{z}_d{}^\top\boldsymbol{q}_i]
\end{bmatrix}
\underbrace{\text{iid}}
\begin{bmatrix}
\mathcal{N}(0,\sigma_1{}^2 d) \\
\cdots \\
\mathcal{N}(0,\sigma_k{}^2 d) \\
\mathcal{N}(0,0) \\
\cdots \\
\mathcal{N}(0,0)
\end{bmatrix}.
$$

We can see that $\sigma_j\boldsymbol{z}_j{}^\top\boldsymbol{q}_i$ for $j <= k$ has very large variance, it means each position has a large probability to get a big positive or negative value. If one position $\sigma_j\boldsymbol{z}_j{}^\top\boldsymbol{q}_i$ for $j <= k$ gets a big positive value, after softmax, the probability in position j for $j > k$ will be almost 0. The position that gets the biggest positive value will almost have a probability 1.0. This completes the proof of malignant entropy collapse. □

E Proof of Weyl's Inequality on Singular Values

Our derivation depends on Horn & Johnson (1991; 2012). Readers can refer to these material for more background information.

Proof. Before we prove Weyl's Inequality on singular values, let us review Courant-Fischer min-max principle that is important for analyzing the singular values of matrix.

Theorem 3 (Courant-Fischer Min-max Principle for Singular Values)
Let $\boldsymbol{W} \in \mathcal{R}^{m\times n}$ be a matrix where $m \geq n$. $\boldsymbol{W}$ has ordered singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. Then, for $i = 1, 2, \ldots, n$, we have

$$\sigma_i(\boldsymbol{W}) = \max_{\substack{S\subset\mathcal{R}^n \\ \dim(S)=i}} \min_{\substack{\boldsymbol{x}\in S \\ \|\boldsymbol{x}\|=1}} \|\boldsymbol{W}\boldsymbol{x}\| = \min_{\substack{S'\subset\mathcal{R}^n \\ \dim(S')=n-i+1}} \max_{\substack{\boldsymbol{x}\in S' \\ \|\boldsymbol{x}\|=1}} \|\boldsymbol{W}\boldsymbol{x}\|$$

where the maximum is taken over all i-dimensional subspaces S of $\mathcal{R}^n$, and the minimum is taken over all unit vectors $\boldsymbol{x}$ in S.

Let $\boldsymbol{W}_1 = \boldsymbol{U}\boldsymbol{\Sigma}_1\boldsymbol{V}^\top$ and $\boldsymbol{W}_2 = \boldsymbol{\mathcal{U}}\boldsymbol{\Sigma}_2\boldsymbol{\mathcal{V}}^\top$ be singular value decompositions of $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ with unitary matrix $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n], \boldsymbol{\mathcal{V}} = [\mathfrak{v}_1, \ldots, \mathfrak{v}_n]$ where $\boldsymbol{v}_i, \mathfrak{v}_i \in \mathcal{R}^n$ and unitary matrix $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m], \boldsymbol{\mathcal{U}} = [\mathfrak{u}_1, \ldots, \mathfrak{u}_m]$, where $\boldsymbol{u}_j, \mathfrak{u}_j \in \mathcal{R}^m$.

Let i and j be positive integers with $1 \leq i, j \leq n$ and $i + j \leq n + 1$. Let $S_1 \equiv \mathrm{Span}\{\boldsymbol{v}_i, \ldots, \boldsymbol{v}_n\}$ and $S_2 \equiv \mathrm{Span}\{\mathfrak{v}_j, \ldots, \mathfrak{v}_n\}$; notice that $\dim(S_1) = n - i + 1$ and $\dim(S_2) = n - j + 1$. Let $k \equiv \dim(S_1 \cap S_2)$, then we have

$$\begin{aligned}\dim(S_1 \cap S_2) &= \dim(S_1) + \dim(S_2) - \dim(S_1 + S_2) = (n - i + 1) + (n - j + 1) - \dim(S_1 + S_2) \\ &\geq (n - i + 1) + (n - j + 1) - n = n - (i + j - 1) + 1 \geq 1.\end{aligned}$$

Because of the bounds assumed for i and j. Thus, the subspace $S_1 \cap S_2$ has positive dimension k, $n - k + 1 \leq i + j - 1$, and we have

$$\begin{aligned}\sigma_{i+j-1}(\boldsymbol{W}_1 + \boldsymbol{W}_2) &\leq \sigma_{n-k+1}(\boldsymbol{W}_1 + \boldsymbol{W}_2) \\ &= \min_{\substack{S\subset\mathcal{R}^n \\ \dim(S)=k}} \max_{\substack{\boldsymbol{x}\in S \\ \|\boldsymbol{x}\|_2=1}} \|(\boldsymbol{W}_1 + \boldsymbol{W}_2)\boldsymbol{x}\|_2 \\ &\leq \max_{\substack{\boldsymbol{x}\in S_1\cap S_2 \\ \|\boldsymbol{x}\|_2=1}} \|(\boldsymbol{W}_1 + \boldsymbol{W}_2)\boldsymbol{x}\|_2 \\ &\leq \max_{\substack{\boldsymbol{x}\in S_1\cap S_2 \\ \|\boldsymbol{x}\|_2=1}} \|\boldsymbol{W}_1\boldsymbol{x}\|_2 + \max_{\substack{\boldsymbol{x}\in S_1\cap S_2 \\ \|\boldsymbol{x}\|_2=1}} \|\boldsymbol{W}_2\boldsymbol{x}\|_2 \\ &\leq \max_{\substack{\boldsymbol{x}\in S_1 \\ \|\boldsymbol{x}\|_2=1}} \|\boldsymbol{W}_1\boldsymbol{x}\|_2 + \max_{\substack{x\in S_2 \\ \|\boldsymbol{x}\|_2=1}} \|\boldsymbol{W}_2\boldsymbol{x}\|_2 = \sigma_i(\boldsymbol{W}_1) + \sigma_j(\boldsymbol{W}_2).\end{aligned}$$

As a special case, the second part of the theorem follows directly from the general result of part (a). Specifically, for $i = j = 1$, we have:

$$\sigma_1(\boldsymbol{W}_1 + \boldsymbol{W}_2) \leq \sigma_1(\boldsymbol{W}_1) + \sigma_1(\boldsymbol{W}_2).$$

This completes the proof. □

F Related Works

Training Dynamics of Transformer. Previous works have delved into understanding the training dynamics of Transformers from two different perspectives: a high-level perspective and a low-level perspective. From a high-level perspective, Scan&Snap (Tian et al., 2023a) unveiled complex phenomena, particularly in single-layer architectures, relating to frequency and discriminative bias. These studies linked sparse attention patterns to token co-occurrence frequencies and observed two-stage behaviors in attention logits. JoMA (Tian et al., 2023b) further improved upon previous models by incorporating residual connections and MLP nonlinearity, analyzing joint training of MLP and self-attention layers, and offering qualitative explanations for multi-layer Transformer dynamics. From a low-level perspective, two critical challenges in Transformer training have been identified: rank collapse (Dong et al., 2021; Noci et al., 2022), where attention output converges to a rank 1 matrix, potentially causing vanishing gradients; and entropy collapse (Zhai et al., 2023), which denotes pathologically low attention entropy, corresponding to highly concentrated attention scores. *In this work, we analyze and prove two different entropy collapse modes and identify the key reason for model failure is spectral energy concentration. Finally, we introduce a simple but effective solution to address this problem.*

Training Stability of Transformer. ReZero (Bachlechner et al., 2021) introduces a simple yet effective mechanism for improving training stability. The key innovation lies in initializing residual connections to zero, which allows networks to learn identity mappings more easily. Admin (Liu et al., 2020) introduces a new network initialization strategy tailored for Transformer to make the network train stable. DeepNorm (Wang et al., 2022) extends the concept of normalization to accommodate increasingly deeper networks. By dynamically adjusting normalization parameters, DeepNorm ensures stability even as network depth increases. LipsFormer Qi et al. (2023a) addresses the specific challenge of stability in transformer networks. By introducing a Lipschitz continuity constraint, Lipsformer effectively mitigates the issue of exploding gradients - a common problem in deep transformer architectures. This approach ensures that the network's output changes smoothly with respect to its input, promoting overall stability. ReZero, Admin and DeepNorm can all be considered as an approach to control the Lipschitz constant of the network in the initial stage. *In this work, by revisiting the training dynamics of Transformer, we can achieve a stable training only by modifying the optimizer instead of using learning rate warmup or changing the network structures as LipsFormer (Qi et al., 2023a) and QKNorm Henry et al. (2020); Dehghani et al. (2023).*

Learning Rate Schedule. Warmup (Loshchilov & Hutter, 2016) has emerged as a must-have technique for ensuring a stable network training, especially in the initial phases of the optimization process. This method involves gradually increasing the learning rate from a small value to the desired initial learning rate over a certain number of training steps or epochs. The cosine learning rate scheduler (Loshchilov & Hutter, 2016) has gained popularity due to its smooth annealing properties. This schedule decreases the learning rate following a cosine curve, starting from an initial value and decaying to a minimum value over a set number of epochs or iterations. Cyclic learning rates (Smith, 2017) involve systematically varying the learning rate between boundary values. The learning rate oscillates between a lower and upper bound, either linearly or following other patterns (e.g., triangular, cosine). The above-mentioned learning rate schedules require specification of a stopping time step T, Defazio et al. (2024) introduces a Schedule-Free approach that avoids the need for this stopping time by eschewing the use of schedules entirely.

Compared to the up-mentioned works, the core novel contributions of our work lie on follows.

1. We present a theoretical analysis for Transformer training and point out two entropy collapse modes, *i.e.*the benign collapse and the malignant collapse.
2. We reveal that *spectral energy concentration (SEC) of* $\boldsymbol{W_q}^\top \boldsymbol{W_k}$ is the main reason of model crash.
3. We introduce AdamW^2, a new optimization strategy motivated by Weyl's Inequality.

G SIMULATION OF THREE ATTENTION MODES

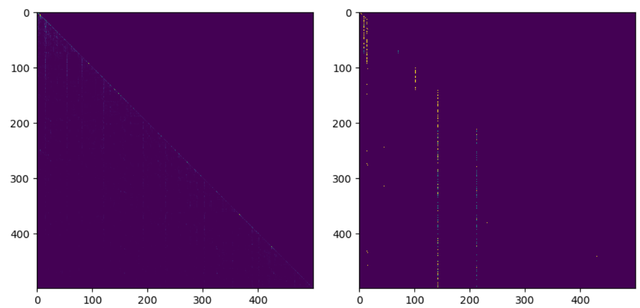
We provide a simple simulation code to simulate three attention modes, but it is important to note that the real picture is more complicated. In real case, in the benign attention entropy mode, ${\bm{W}_q}^\top \bm{W}$ is a non-symmetric positive quasi-definite square matrix instead of a symmetric positive definite matrix in our simulation. The code is just to demonstrate the core ideas behind three attention modes.

CODE 1: Simulation of Three Attention Modes.

```
import torch
import torch.nn
import matplotlib.pyplot as plt
import numpy as np


#Randomly generate data and weight matrices
d_q, d, num_tokens= 64, 768, 197
Wq = torch.randn(d_q, d)
Wk = torch.randn(d_q, d)
X = torch.randn(d, num_tokens)
W = torch.mm(Wq.T, Wk)


# Normal attention mode
W1 = W
P = torch.mm(torch.mm(X.T, W1), X)
attn_map1 = P.softmax(dim=1)


# Malignant attention entropy collapse mode
u,s,v = torch.svd(W)
s[0:3] = torch.tensor([3., 2., 1.])*s[0:3]
s[3:]  = 0.0
W2 = torch.mm(torch.mm(u, torch.diag(s)), v.T)
P = torch.mm(torch.mm(X.T, W2), X)
attn_map2 = P.softmax(dim=1)


# Benign attention entropy collapse mode
u,s,v = torch.svd(W)
W3 = torch.mm(torch.mm(u, torch.diag(s)), u.T)
P = torch.mm(torch.mm(X.T, W3), X)
attn_map3 = P.softmax(dim=1)


# Plot figures
fig, axs = plt.subplots(1, 3, figsize=(15, 5))
axs[0].imshow(attn_map1.detach().numpy())
axs[1].imshow(attn_map2.detach().numpy())
axs[2].imshow(attn_map3.detach().numpy())
plt.show()
```

H Attention Map Visualization of GPT

Figure 7 visualizes the dynamic process of attention map as the number of training steps increases for a successful and unsuccessful GPT-Small model. It should be noted that the GPT model uses a lower triangular attention mask.



(a) *Block 11 (successful).* (b) *Block 11 (unsuccessful).*

Figure 7: Visualization of the dynamic process of attention map as the number of training steps increases for a successful and unsuccessful GPT-Small model. Attention map gradually becomes sparse and low-rank along with the training process in a failure case. *Please click the images to play the flash. Best viewed with Acrobat Reader.*

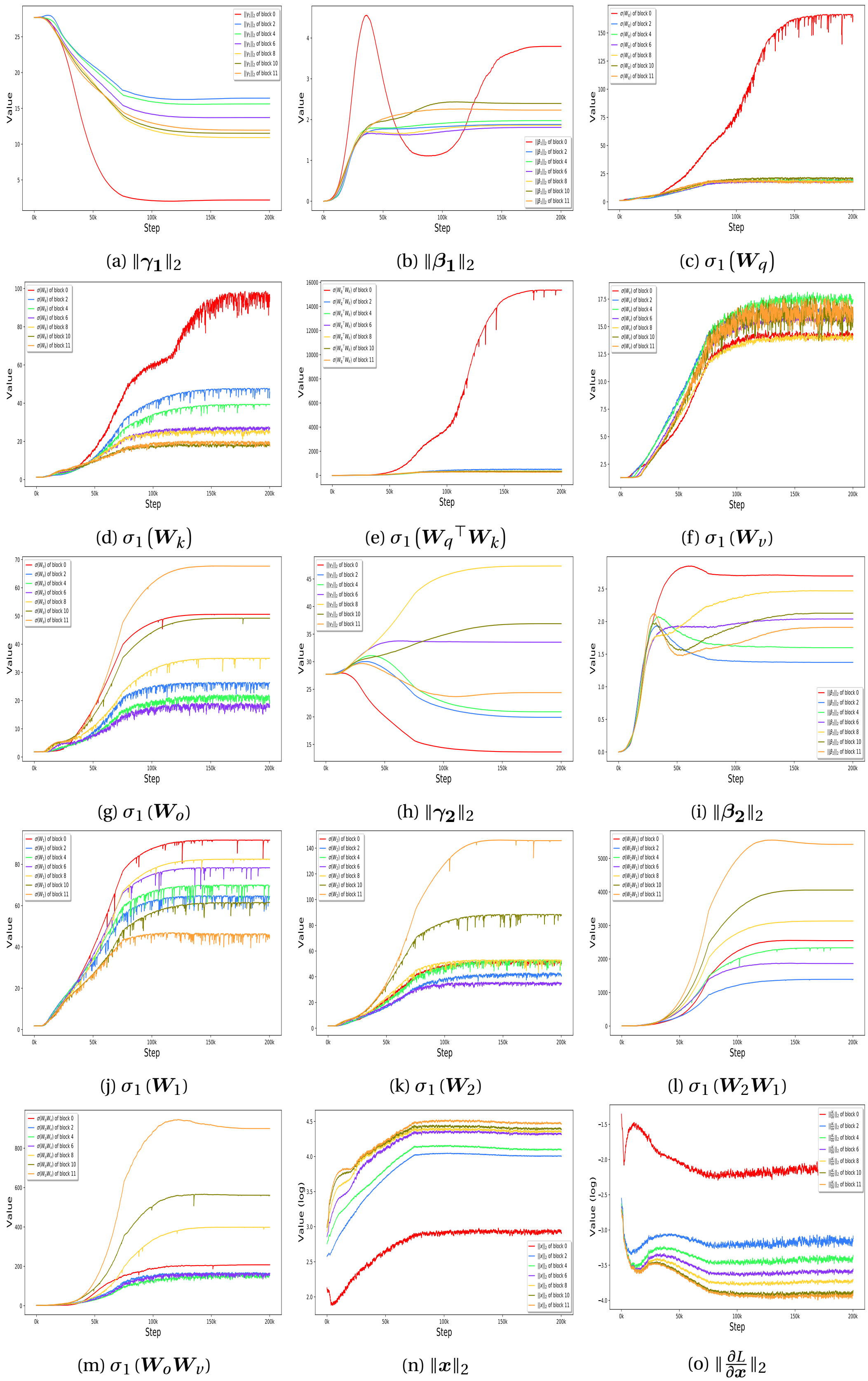
In Figure 7, the attention values in a successful case distribute to different position, but the attention values in a unsuccessful case will only concentrate into several directions.

I More Training Dynamics of ViT and GPT

Figure 8 visualizes a successful ViT training process. Compared with Figure 1, we find several significant differences as follows.

- In a successful ViT training process, the value of $\sigma_1(\boldsymbol{W_q}^\top \boldsymbol{W}_k)$ increases to 16,000, then starts to oscillate smoothly. But for an unsuccessful training, the value suddenly increases to a very large value, around 300,000, it triggers the model crash,
- The γ_1 and β_1 in a successful ViT training process are very smooth, but they change a lot in an unsuccessful case,
- The fast increase of $\sigma_1(\boldsymbol{W_q}^\top \boldsymbol{W}_k)$ is accompanied by a fast increase of $\boldsymbol{W}_q$ and $\boldsymbol{W}_k$.

We can observe similar phenomenon in Figure 9 and Figure 15. In a successful GPT training process, the value of $\sigma_1(\boldsymbol{W_q}^\top \boldsymbol{W}_k)$ increases to 60, then starts to oscillate smoothly. But for an unsuccessful GPT training, the value increases to 20,000. The difference between the sclae of value between GPT and ViT may be due to the density and sparsity of the supervision signal. In GPT, each token will contribute a gradient, but in ViT, only one class label in an image provides a supervision information.



(a) $\|\boldsymbol{\gamma_1}\|_2$ (b) $\|\boldsymbol{\beta_1}\|_2$ (c) $\sigma_1(\boldsymbol{W}_q)$

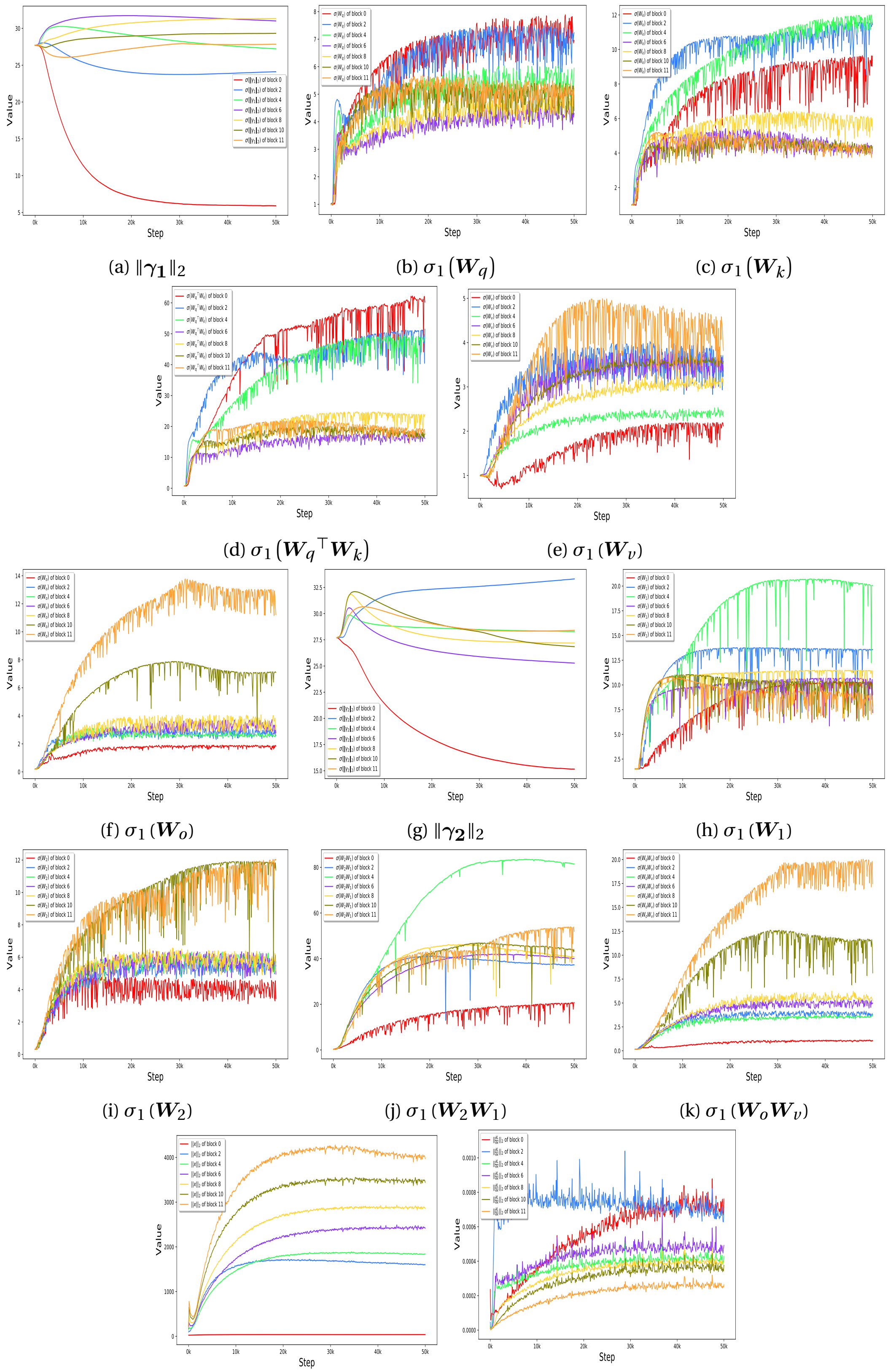
(d) $\sigma_1(\boldsymbol{W}_k)$ (e) $\sigma_1(\boldsymbol{W}_q{}^\top \boldsymbol{W}_k)$ (f) $\sigma_1(\boldsymbol{W}_v)$

(g) $\sigma_1(\boldsymbol{W}_o)$ (h) $\|\boldsymbol{\gamma_2}\|_2$ (i) $\|\boldsymbol{\beta_2}\|_2$

(j) $\sigma_1(\boldsymbol{W}_1)$ (k) $\sigma_1(\boldsymbol{W}_2)$ (l) $\sigma_1(\boldsymbol{W}_2\boldsymbol{W}_1)$

(m) $\sigma_1(\boldsymbol{W}_o\boldsymbol{W}_v)$ (n) $\|\boldsymbol{x}\|_2$ (o) $\|\frac{\partial L}{\partial \boldsymbol{x}}\|_2$

FIGURE 8: Training dynamics of a successful ViT training.



(a) $\|\boldsymbol{\gamma_1}\|_2$ (b) $\sigma_1(\boldsymbol{W}_q)$ (c) $\sigma_1(\boldsymbol{W}_k)$

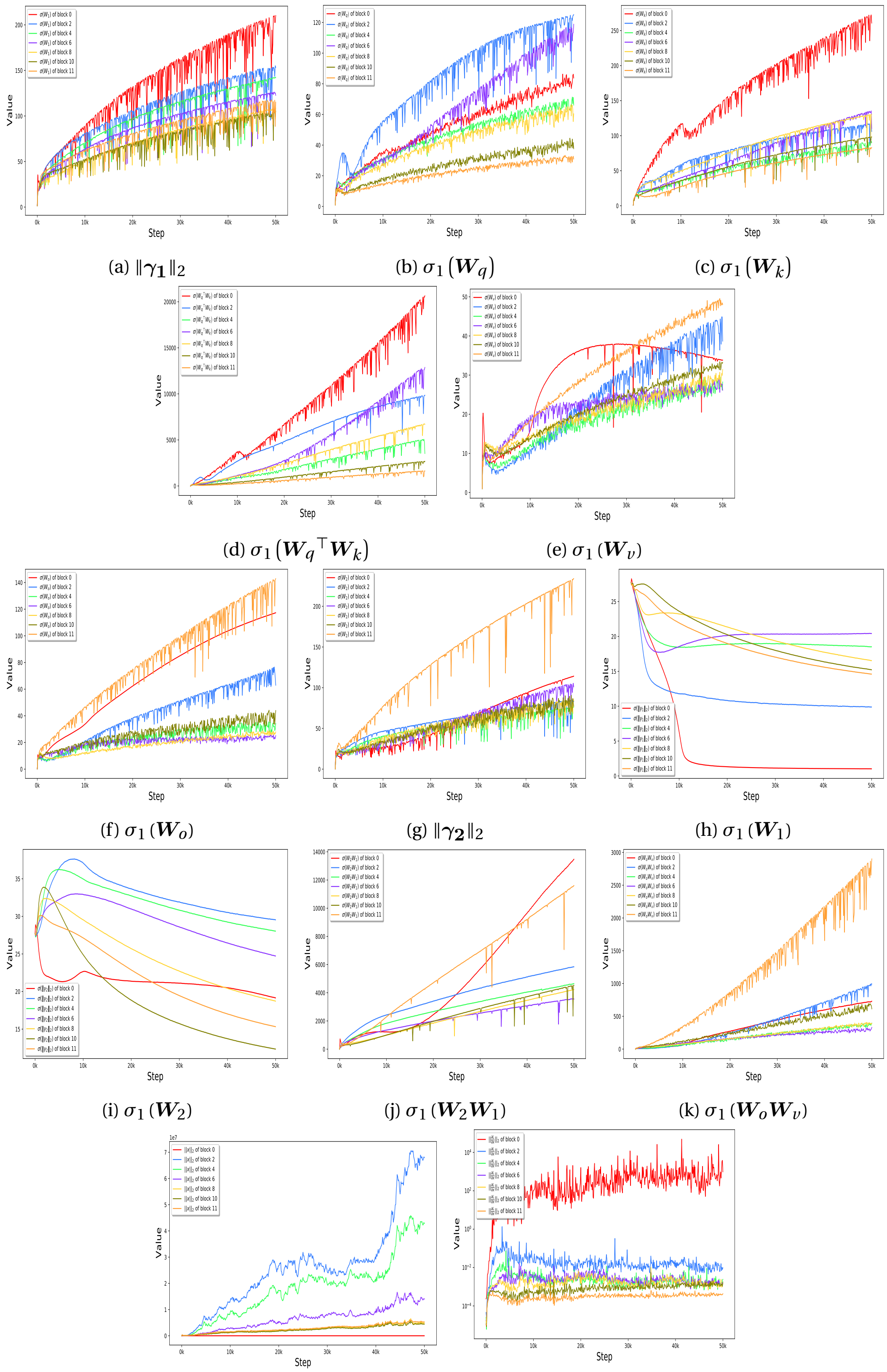
(d) $\sigma_1(\boldsymbol{W}_q{}^\top \boldsymbol{W}_k)$ (e) $\sigma_1(\boldsymbol{W}_v)$

(f) $\sigma_1(\boldsymbol{W}_o)$ (g) $\|\boldsymbol{\gamma_2}\|_2$ (h) $\sigma_1(\boldsymbol{W}_1)$

(i) $\sigma_1(\boldsymbol{W}_2)$ (j) $\sigma_1(\boldsymbol{W}_2\boldsymbol{W}_1)$ (k) $\sigma_1(\boldsymbol{W}_o\boldsymbol{W}_v)$

(l) $\|\boldsymbol{x}\|_2$ (m) $\|\frac{\partial L}{\partial \boldsymbol{x}}\|_2$

FIGURE 9: Training dynamics of a successful GPT training.



(a) $\|\boldsymbol{\gamma_1}\|_2$ (b) $\sigma_1\left(\boldsymbol{W}_q\right)$ (c) $\sigma_1\left(\boldsymbol{W}_k\right)$

(d) $\sigma_1\left(\boldsymbol{W}_q{}^{\top}\boldsymbol{W}_k\right)$ (e) $\sigma_1\left(\boldsymbol{W}_v\right)$

(f) $\sigma_1\left(\boldsymbol{W}_o\right)$ (g) $\|\boldsymbol{\gamma_2}\|_2$ (h) $\sigma_1\left(\boldsymbol{W}_1\right)$

(i) $\sigma_1\left(\boldsymbol{W}_2\right)$ (j) $\sigma_1\left(\boldsymbol{W}_2\boldsymbol{W}_1\right)$ (k) $\sigma_1\left(\boldsymbol{W}_o\boldsymbol{W}_v\right)$

(l) $\|\boldsymbol{x}\|_2$ (m) $\|\frac{\partial L}{\partial \boldsymbol{x}}\|_2$

FIGURE 10: Training dynamics of an unsuccessful GPT training.

J EXPERIMENT OF 1B VIT

To further evaluate the effectiveness of our method at a larger scale, we assessed ViT-g with 1B parameters. The ViT-g model architecture consists of 40 layers with a hidden dimension of 1408, 16 attention heads, and an MLP dimension of 6144. The total parameter count is 1011M, around one billion parameters. We conducted a comparative study between ViT-g with AdamW^2 and ViT-g with AdamW, where ViT-g with AdamW was evaluated under two settings: with and without learning rate warmup. Our AdamW^2 does not use warmup. The comparison results are presented in Figure 11 and Figure 12.

Figure 11 shows that ViT-g with AdamW crashes after only a few training steps when running without warmup. While the use of warmup enables ViT-g to complete training, but the loss spikes one time. Our ViT-g with AdamW^2 not only achieves stable training without warmup but also demonstrates better performance.

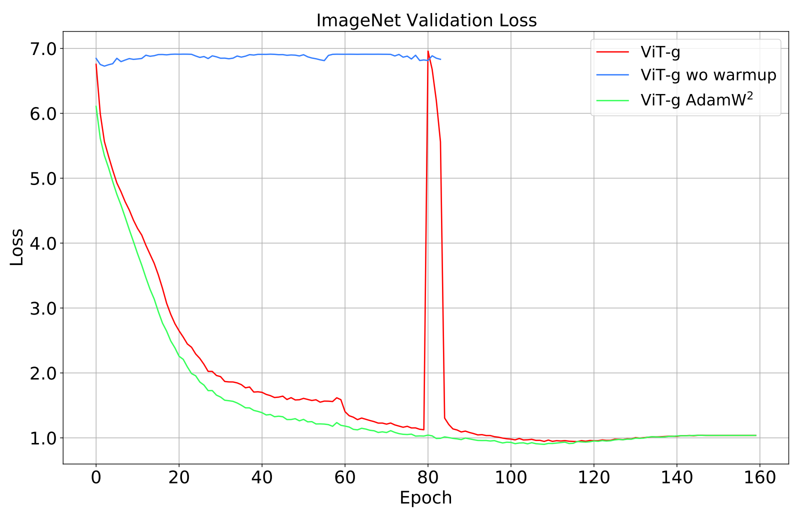


FIGURE 11: Comparison of loss curve of AdamW^2 and AdamW on ViT-g model.

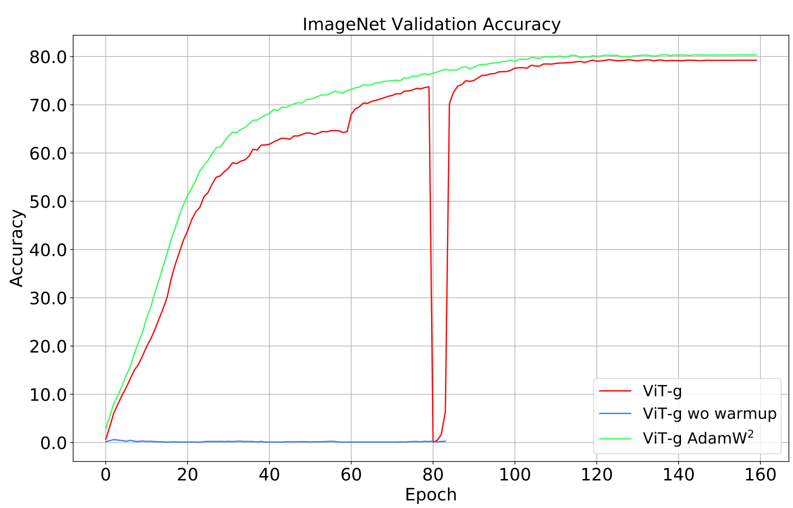


FIGURE 12: Comparison of accuracy of AdamW^2 and AdamW on ViT-g model.

K EXPERIMENT OF 774M NANOGPT

We also evaluated the effectiveness of our method on a larger-scale language model, termed as nanoGPT-large. The model architecture consists of 36 layers with a hidden dimension of 1280 and 20 attention heads. The total parameter count is 774M. Our experimental setup strictly follows the nanoGPT configuration, including all learning rate settings. It is important to note that training nanoGPT-large is computationally intensive, requiring two weeks to train 600K steps on 16 A800 GPUs. To reduce the training time, we limited our training to 100K steps instead of the full 600K steps. The comparison results are presented in Figure 13. We can see from Figure 13, nanoGPT-large achieves a stable training without warmup and obtains a similar validation loss with its counterpart, GPT2-large. This further verifies our understanding to the model crash of Transformer.

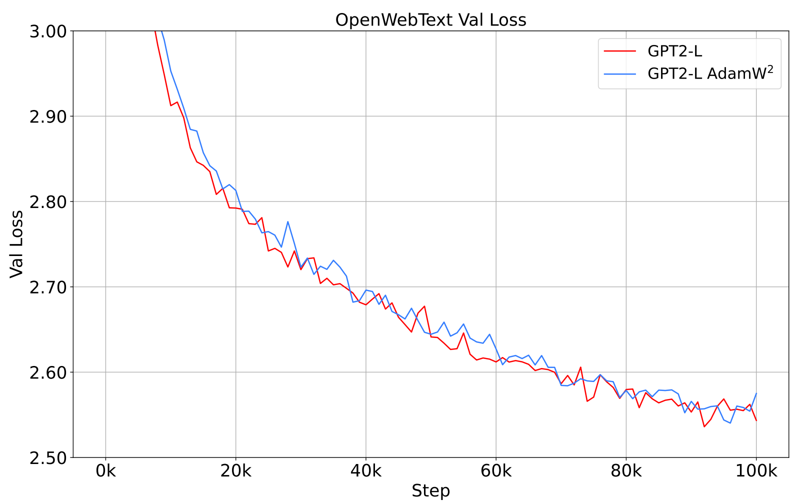


FIGURE 13: Comparison of validation loss of AdamW^2 and AdamW on nanoGPT-large model.

L EXPERIMENT OF FLATTEN-SWIN

Besides ViT, GPT, and Swin-Transformer, we further validated our approach on Flatten-Transformer Han et al. (2023). We used Flatten-Swin, and we compared the performance of our method and the baseline method training 150 and 300 epochs. Our method could stably train and demonstrate performance comparable to the baseline. This further verified the correctness of our understanding of neural network stability.

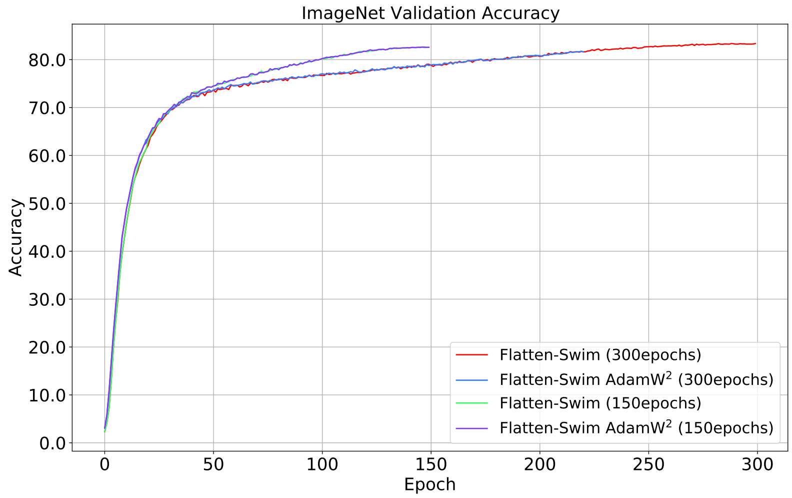
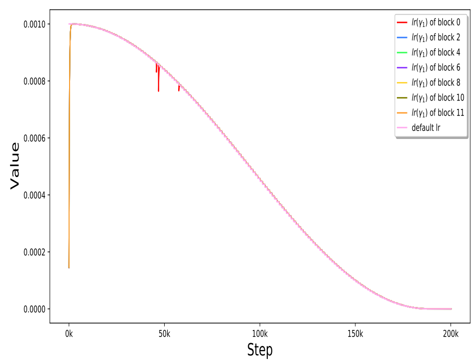


FIGURE 14: Evaluation of Flatten-Swin.

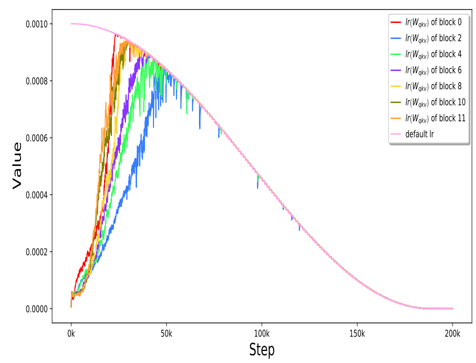
M Actual Learning Rate Curve along with Training steps

We recorded the actual learning rate throughout the training steps, we sample one point every 50 steps. Our initial setting of the learning rate is a cosine learning rate scheduler without warmup. If $\alpha_t \frac{\sigma_1(\nabla \boldsymbol{W}_t)}{\sigma_1(\boldsymbol{W}_{t-1})} > \tau$, then α_t will be truncated to $\tau \frac{\sigma_1(\boldsymbol{W}_{t-1})}{\sigma_1(\nabla \boldsymbol{W}_t)}$. From the figure, we observe that $\boldsymbol{\gamma_1}$ and $\boldsymbol{\gamma_2}$ in RMSNorm only exceed the preset τ during the initial training phase and rarely exceed it afterwards. For other curves, they somewhat look like a curve with learning rate warmup, but we can see that different blocks have different learning rates.

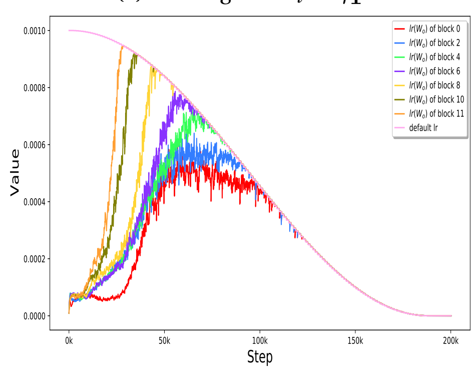
We also observe that shallower layers are more likely to violate the preset τ value. It means the shallower layers are more likely to lead to a greater update of weight matrix and typically require a smaller learning rate. Additionally, we notice that for the weight matrix $\boldsymbol{W}_2$, it is more prone to exceeding the preset τ value compared to the weight matrix $\boldsymbol{W}_1$.



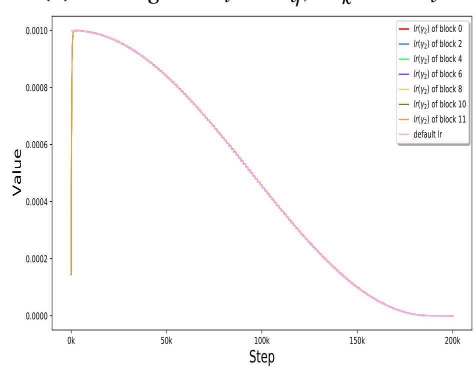
(a) learning rate α_t of $\boldsymbol{\gamma_1}$



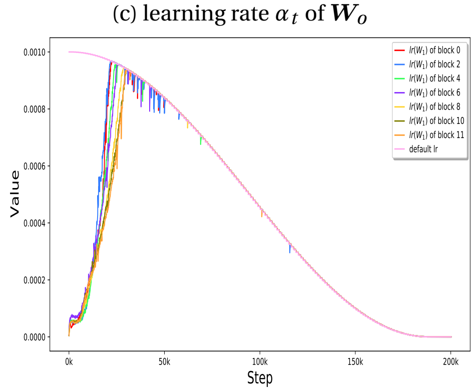
(b) learning rate α_t of $\boldsymbol{W}_q$, $\boldsymbol{W}_k$ and $\boldsymbol{W}_v$



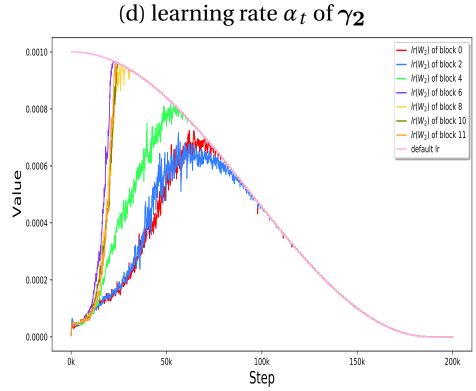
(c) learning rate α_t of $\boldsymbol{W}_o$



(d) learning rate α_t of $\boldsymbol{\gamma_2}$



(e) learning rate α_t of $\boldsymbol{W}_1$



(f) learning rate α_t of $\boldsymbol{W}_2$

Figure 15: Actual Learning Rate Curve along with Training steps.

N TRAINING CONFIGURATIONS

Training Configurations. We list the training configurations of ViT, GPT, Swin-Transformer and Flatten-Swin in Table 2. For ViT, GPT, Swin-Transformer and Flatten-Swin, we do not use learning rate warmup. For GPT, we follow the experimental configurations of nanoGPT (Karpathy, 2022), all parameters are same as GPT2 (Radford et al., 2019). For ViT, we use Timm (Wightman, 2019). For Swin-Transformer, we use the original code provided by Liu et al. (2021). For Flatten-Swin, we use the original code provided by (Han et al., 2023).

TABLE 2: Training configurations for ViT, GPT and Swin-Transformer.

(a) Training configurations for ViT.

training config	ViT-B/L/g (224^2)
optimizer	AdamW^2
τ (In default)	0.004 or 0.003
warmup epochs	0
weight init	Truncated Xavier
base learning rate	1e-3
weight decay	0.05/0.1
optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.99$
batch size	1024
training epochs	150
learning rate schedule	cosine decay
randaugment	$(9, 0.5)$
mixup	0.8
cutmix	1.0
random erasing	0
label smoothing	0.1
stochastic depth	0.1/0.5
gradient clip	None
exp. mov. avg. (EMA)	no

(b) Training configurations for GPT.

training config	GPT-S/L
optimizer	AdamW^2
τ	0.01
warmup epochs	0
weight init	Xavier
baseline learning rate	0.0006 or 0.00025
weight decay	0.1
optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.95$
tokens seen each update	500,000
max iters	600K or 100K
batch size	480
sequence length	1024
dropout	0.0
bfloat16	True
gradient clipping	1.0

(c) Training configurations for Swin-Transformer.

training config	Swin S/B (224^2)
optimizer	AdamW^2
τ (In default)	0.004
warmup epochs	0
training epochs	300
others	same as Liu et al. (2021)

(d) Training configurations for Flatten-Swin.

training config	Flatten-Swin S (224^2)
optimizer	AdamW^2
τ (In default)	0.004
warmup epochs	0
training epochs	150 or 300
others	same as Han et al. (2023)

O NON-SYMMETRIC POSITIVE QUASI-DEFINITE SQUARE MATRIX

When we mention a non-symmetric positive quasi-definite square matrix, we mean it has the following three properties,

1. $\boldsymbol{W}_q^\top \boldsymbol{W}_k$ is not symmetric because generally, $\boldsymbol{W}_q^\top \boldsymbol{W}_k \neq \boldsymbol{W}_k^\top \boldsymbol{W}_q$,
2. $\boldsymbol{W}_q^\top \boldsymbol{W}_k$ is a square matrix and most of its eigenvalues are larger than 0, and only very few are less than 0.0. So we call it positive quasi-definite matrix.
3. if we assume $\boldsymbol{W} = \boldsymbol{W}_q^\top \boldsymbol{W}_k$ is positive definite matrix, if for each element in $\boldsymbol{x}$ is sampled from a standard Gaussian distribution, we can prove

$$\mathbb{E}\left[\boldsymbol{x}_i{}^\top \boldsymbol{W} \boldsymbol{x}_i\right] \gg \mathbb{E}\left[\boldsymbol{x}_i{}^\top \boldsymbol{W} \boldsymbol{x}_j\right]$$

when $i \neq j$, see Appendix C for the proof.

P DISCUSSION ABOUT RANK COLLAPSE, ENTROPY COLLAPSE AND SPARSE YET LOW-RANK ENTROPY MATRIX

Before we start our discussion, let us see three matrices,

$$\boldsymbol{A} = \begin{pmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{pmatrix}, \boldsymbol{B} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \boldsymbol{C} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

We can see that $\boldsymbol{A}$ is low-rank, $\boldsymbol{B}$ is sparse but not low-rank, $\boldsymbol{C}$ is sparse and low-rank.

In previous papers (Dong et al., 2021; Zhai et al., 2023), researchers have analyzed the problem of model crash via *rank collapse of activations and entropy collapse of attention map*. Dong et al. (Dong et al., 2021) attributes the model crash into *rank collapse of the activations*, but Zhai et al. (2023) think it is the *entropy collapse of the attention map* leading to the model crash.

However, based on our analysis, we can find a counterexamples for entropy collapse, and meanwhile the rank collapse of the activation cannot fully describe the inner reason of the model crash (the weight matrix instead of activations). When the state of $\boldsymbol{C}$ usually happens, the model crashed,

- Rank collapse of the activations cannot reveal the underlying cause that exists in the weight matrix. Weight matrix is the inner key ingredient of the model instead of activations.
- $\boldsymbol{B}$ is a counterexample of entropy collapse. we observe that in some successful cases, state $\boldsymbol{B}$ occurs. According to the definition of entropy collapse, state $\boldsymbol{B}$ should lead to model crash; however, our experiments show that the model remains stable in this state.
- Sparse yet low-rank attention matrix is the state of the attention map when a model crashs. We believe rank collapse of activations and entropy collapse of attention map are not enough to describe the state of the model crash precisely. According to our analysis, the Spectral Energy Concentration (SEC) of the $\boldsymbol{W}_q^\top \boldsymbol{W}_k$ is the inner reason the model crash, and the sparse yet low-rank attention matrix is the phenomena observed on the attention matrix.

In summary, our paper, via a rigid theoritical analysis, our paper reveals the Spectral Energy Concentration (SEC) of the $\boldsymbol{W}_q^\top \boldsymbol{W}_k$ is the inner reason the model crash, and the sparse yet low-rank attention matrix is the phenomena that is observed on the attention matrix.