Preconditioned Attention: Enhancing Efficiency in Transformers

Hemanth Saratchandran
Australian Institute for Machine Learning (AIML), Adelaide University
hemanth.saratchandran@adelaide.edu.au

Abstract

Central to the success of Transformers is the attention block, which effectively models global dependencies among input tokens associated to a dataset. However, we theoretically demonstrate that standard attention mechanisms in transformers often produce ill-conditioned matrices with large condition numbers. This ill-conditioning is a well-known obstacle for gradient-based optimizers, leading to inefficient training. To address this issue, we introduce preconditioned attention, a novel approach that incorporates a conditioning matrix into each attention head. Our theoretical analysis shows that this method significantly reduces the condition number of attention matrices, resulting in better-conditioned matrices that improve optimization. Conditioned attention serves as a simple drop-in replacement for a wide variety of attention mechanisms in the literature. We validate the effectiveness of preconditioned attention across a diverse set of transformer applications, including image classification, object detection, instance segmentation, long sequence modeling and language modeling.

1 Introduction

Transformers have revolutionized machine learning, emerging as the dominant architecture for a diverse range of applications, including natural language processing, computer vision, and robotics. Introduced by Vaswani et al. (2017), transformers have redefined how models process and interpret data by capturing both local and global dependencies. Central to this innovation is the self-attention mechanism, a key component that computes pairwise interactions among all input tokens simultaneously. By dynamically weighing the relative importance of tokens, self-attention enables transformers to model intricate patterns and complex relationships within data, establishing them as a foundational architecture for modern machine learning applications.

In this paper, we focus on the conditioning of the self-attention matrix. The conditioning of a matrix is measured by its condition number, defined as the ratio of its largest to smallest singular value. It is well-established that poorly conditioned matrices (i.e. those with high condition numbers) can adversely impact the convergence of gradient-based optimization algorithms, making training challenging (Nocedal & Wright, 1999). While prior studies have examined the conditioning of feedforward layers in neural networks (Agarwal et al., 2021; Liu et al., 2022; Saratchandran et al., 2025b), the conditioning of self-attention mechanisms in transformers remains largely unexplored.

We introduce a theoretical framework to analyze the condition number of the self-attention matrix and show that it can be bounded by a quantity dependent on the Frobenius norm of the values matrix comprising the self-attention mechanism. Leveraging this insight, we propose a method to reduce the condition number of the self-attention matrix. Drawing inspiration from numerical methods, we employ a preconditioner—an auxiliary matrix designed to lower the conditioning of the target matrix. However, constructing an effective preconditioner is often challenging, as it requires tailoring to the specific structure of the problem. Leveraging the unique structure of the self-attention matrix, we propose a novel preconditioner that significantly reduces its condition number. This approach, which we term preconditioned attention, improves the conditioning of the self-attention mechanism and lays the groundwork for more stable and efficient training of transformers. Figure 1 presents a schematic representation of a preconditioned attention layer. In this design, the self-attention matrix is multiplied on the left by a

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

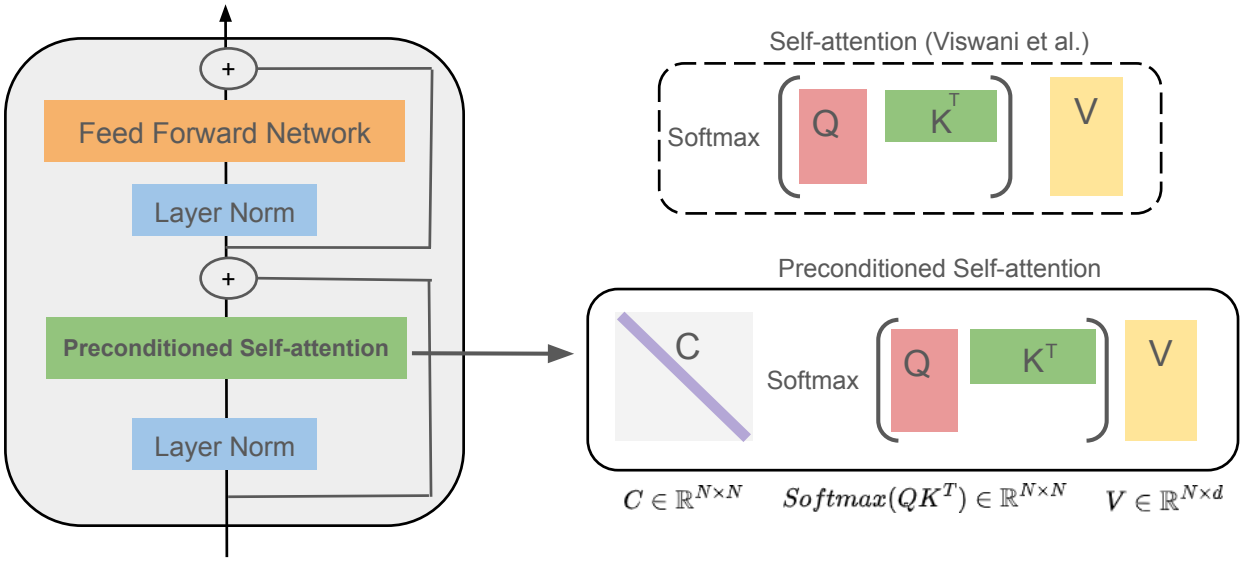

Figure 1: Schematic representation of preconditioned self-attention. Left: A layer of a general transformer employing a preconditioned self-attention block. Right: Self-attention (top) and preconditioned self-attention (bottom) are compared. The key difference is that preconditioned self-attention applies a multiplication by a diagonal preconditioner matrix C, which depends on the query Q, key K, and value V.

square diagonal matrix C, which serves to condition the self-attention matrix, effectively reducing its condition number. The matrix C is dynamically computed based on the queries Q, keys K, and values V of the self-attention matrix.

We demonstrate the effectiveness of preconditioned self-attention across a diverse range of applications, including image classification, object detection, instance segmentation, text classification, and physics-based modeling. In each case, we show that preconditioned self-attention serves as a robust and efficient replacement for various attention mechanisms commonly used in the literature, consistently achieving superior results across all these tasks. One of the key strengths of preconditioned self-attention lies in its versatility. It is agnostic to the underlying attention mechanism, making it compatible with a wide range of architectures. As we demonstrate, it can serve as a drop in replacement to advanced attention mechanisms proposed in recent works (Ali et al., 2021; Liu et al., 2021; Touvron et al., 2021; Yuan et al., 2022; Ding et al., 2022; Xiong et al., 2021b), offering a straightforward yet powerful enhancement to existing models. Our main contributions are:

1. A theoretical framework demonstrating that self-attention mechanisms are inherently ill-conditioned, along with the introduction of preconditioned self-attention—a novel approach that employs a preconditioner to modify the self-attention matrix and significantly reduce its condition number.
2. An empirical evaluation of preconditioned attention across various transformer architectures and diverse applications from the literature, demonstrating the effectiveness of our proposed methodology.

2 Related Work

Efficient Attention. In recent years, numerous methods have been proposed to improve the efficiency and effectiveness of transformers, particularly by addressing their computational complexity and rethinking attention mechanisms. The Data-efficient Image Transformer (DeiT) (Touvron et al., 2021) focuses on reducing data requirements by utilizing distillation tokens, which significantly enhance the training efficiency and performance of vision transformers without relying on large-scale datasets. Cross-Covariance Image Transformer (XCiT) (Ali et al., 2021) introduces a novel approach to attention, operating on the cross-covariances of spatial features, thereby enabling efficient interactions across spatial dimensions and reducing computational costs. Vision Outlooker (VOLO) (Yuan et al., 2022) refines the attention mechanism by incorporating an outlook attention module, designed to capture long-range dependencies while maintaining computational efficiency, leading to superior performance in vision tasks. Meanwhile, Nyströmformer (Xiong et al., 2021b) employs a Nyström-based approximation for the standard self-attention mechanism, reducing its quadratic

complexity to near-linear time while retaining the core properties of attention. This transformer has shown great results in modeling long range sequence modeling. We will show that our framework of conditioning self-attention can be added as a drop in replacement to many of these newer forms of attention yielding superior results.

Conditioning. A long line of research has examined neural network training through the perspective of conditioning. Early analyses framed the problem in terms of the neural tangent kernel (NTK), showing that its spectral properties largely determine optimization behavior in the infinite-width limit. In particular, well-conditioned NTKs were linked to more stable convergence (Jacot et al., 2018; Liu et al., 2022). Beyond kernel analyses, architectural choices have also been identified as critical for conditioning Saratchandran et al. (2024); Zheng et al. (2025); MacDonald et al. (2023); Chng et al. (2025); Saratchandran et al. (2025a); Saratchandran & Lucey (2025, 2026); Qin et al. (2022). Depth, for instance, has been shown to enhance the conditioning of gradient updates, facilitating more effective optimization (Agarwal et al., 2021). In the context of Transformers, skip connections have been highlighted as an intrinsic mechanism that regularizes conditioning and prevents instabilities in very deep networks (Ji et al., 2025b,a). Furthermore, works in fine-tuning have shown how to better the conditioning of low-rank adapters for better training Ji et al. (2024); Albert et al. (2025a,b).

3 Notation

In this section, we formally define the transformer architecture by describing the structure of a transformer layer, along with notation for various mathematical elements that will be referenced in subsequent sections. For further foundational details on transformers, readers may consult Vaswani et al. (2017).

A layer in a transformer can be represented as a mapping

$$\mathbf{T} : \mathbb{R}^{N\times D} \to \mathbb{R}^{N\times D} \tag{1}$$

defined by

$$\mathbf{T}(X) = \mathbf{F}(\mathbf{A}(X) + X), \tag{2}$$

where $\mathbf{F}$ denotes a feed forward network with a residual connection, and $\mathbf{A}$ represents an attention mechanism.

Self-attention is composed of three learnable matrices—query (Q), key (K), and value (V)—defined for an input sequence $X \in \mathbb{R}^{N\times D}$ as follows: $q = XQ$, $k = XK$, and $v = XV$, where $Q, K \in \mathbb{R}^{D\times d}$ and $V \in \mathbb{R}^{D\times d}$. The output of the attention head $\mathbf{A}(X)$ is then given by

$$\mathbf{A}(X) = \mathbf{softmax}(qk^T)v, \tag{3}$$

where **softmax** is the softmax activation that acts row-wise on the matrix qk^T. Note that then $A(X) \in \mathbb{R}^{N\times d}$.

In general, multiple attention heads $\mathbf{A}_i$ for $1 \leq i \leq h$ are utilized, each of dimension $N \times d_i = N \times \frac{d}{h}$. These are then concatenated to produce a multi-head attention output,

$$\mathbf{A} = [\mathbf{A}_1, \cdots, \mathbf{A}_h], \tag{4}$$

which is subsequently fed into the feedforward layer. The full transformer architecture is obtained by sequentially stacking several such transformer layers, as defined in Equation (2).

Given an arbitrary matrix M we will denote the Frobenius norm of M by $||M||_F$. If M is of size $n \times d$ and $k = \min\{n, d\}$ then we will denote the singular values of M by $\sigma_1, \ldots, \sigma_k$ and order them so that $\sigma_1 \geq \cdots \geq \sigma_k$. We remind the reader that the following formula exists $||M||_F = \sqrt{\sigma_1^2 + \cdots + \sigma_k^2}$. We will use the standard terminology SVD to denote the singular value decomposition of a matrix. In our main theorem we will need to consider the rows of a matrix M. We will denote the i-th row of M by $M_{i,}$, where if M has size $n \times d$ then each $M_{i,}$ is a vector of shape $1 \times d$ for each $1 \leq i \leq n$. The 2-norm of this vector will be denoted by $||M_{i,}||_2$.

4 Theoretical Framework

4.1 Conditioning of self-attention

In this section we study the conditioning of the self-attention matrix of a transformer.

Definition 4.1. Let A be an $n \times d$ matrix of full rank. The condition number of A, denoted by κ, is defined as

$$\kappa(A) = \frac{\sigma_1}{\sigma_k} \tag{5}$$

where $k = \min\{n, d\}$, and σ_1 denotes the largest singular value of A and σ_k the smallest singular value of A. Note that as A is assumed full rank $\sigma_k \neq 0$ and the condition number is well-defined. Furthermore, observe that $\kappa(A) \geq 1$ since $\sigma_1 \geq \sigma_k$.

The condition number plays a crucial role in iterative algorithms, such as gradient descent, as matrices with lower condition numbers within a non-linear system lead to better convergence of these algorithms Nocedal & Wright (1999); Agarwal et al. (2021); Liu et al. (2022); Saratchandran et al. (2025b).

In general, for large matrices computing the singular value decomposition of the matrix each time to compute

the condition number is extremely difficult. The following estimate of Guggenheimer et al. Guggenheimer et al. (1995) gives a useful bound on the condition number of a full rank matrix which can serve as a good measure of how complex the condition number can be.

Theorem 4.2 (Guggenheimer et al. (1995))**.** *Let A be an $n \times d$ matrix of full rank. Let $k = \min\{n, d\}$. Then the condition number of A has the following bound*

$$\kappa(A) \leq \frac{2}{\sigma_1 \cdots \sigma_k} \left(\frac{||A||_F}{\sqrt{k}} \right)^k. \tag{6}$$

Our first main insight is that the above theorem gives a bound on the condition number of self-attention that depends on the values matrix v of the self-attention $\mathbf{softmax}(qk^t)v$.

Theorem 4.3. *Let $A = \mathbf{softmax}(qk^T)v$ be the standard self-attention of size $n \times d$ and assume A is of full rank. Let $\sigma_1 \geq \cdots \geq \sigma_k$ denote the singular values of A, where $k = \min\{n, d\}$. Then*

$$\kappa(A) \leq \frac{2}{\sigma_1 \cdots \sigma_k} \left(\frac{\sqrt{n}}{\sqrt{k}} ||v||_F \right)^k \tag{7}$$

The proof of Theorem 4.3 is given in Section A. We observe that if $||v||_f > 1$ then since $n \geq k$ the term $\left(\frac{\sqrt{n}}{\sqrt{k}} ||v||_F \right)^k$ will grow exponentially in k. Thus for large attention matrices of full rank this term can be extremely large leading to a high condition number.

4.2 A preconditioner for self-attention

In Theorem 4.3 we saw that the self-attention matrix in a transformer layer can be bounded by a quantity that depended exponentially on the Frobenius norm of the values matrix. In this section, we would like to design a matrix C so that when we consider the product

$$C \cdot \mathbf{softmax}(qk^T)v \tag{8}$$

it satisfies

$$\kappa(C \cdot \mathbf{softmax}(qk^T)v) \leq \kappa(\mathbf{softmax}(qk^T)v). \tag{9}$$

In other words, we want to understand whether there is a matrix C that we can use to multiply the self-attention matrix $\mathbf{softmax}(qk^T)v$ with so as to reduce the condition number. Such a C is called a preconditioner.

In general, computing the condition number of a matrix directly via the SVD is a costly process. However, Theorem 4.2 provides a useful upper bound that can serve as a complexity measure of the condition number of a matrix without needing to directly access the SVD.

Algorithm 1: Forward Pass: Preconditioned Attention

Input: Queries q, Keys k, Values v
Output: Output representation y

1. Attention weights: $A \leftarrow \mathbf{softmax}(qk^\top)$.
2. Construct the preconditioner C as defined in Theorem 4.4.
3. Apply the preconditioner: $\tilde{A} \leftarrow C \cdot A$.
4. Compute the output: $y \leftarrow \tilde{A} \cdot v$.

We define this complexity measure of a matrix A with rank k by

$$\mu(A) := \frac{2}{\sigma_1 \cdots \sigma_k} \left(\frac{||A||_F}{\sqrt{k}} \right)^k. \tag{10}$$

Our goal will be to reduce the quantity $\mu(A)$. The following theorem produces a matrix C that when multiplied with a self-attention matrix brings down the quantity $\mu(A)$.

Theorem 4.4. *Let $A = \mathbf{softmax}(qk^T)v$ denote a self-attention matrix, of size $n \times d$, within a transformer layer and assume A has full rank. Let C be defined as the $n \times n$ diagonal matrix whose i-th diagonal entry is given by $\frac{1}{||A_i,||_2}$. Then*

$$\mu(C \cdot A) \leq \mu(A). \tag{11}$$

The proof of Theorem 4.4 is given in Section A. Theorem 4.4 motivates the consideration of a new type of attention mechanism defined by the equation

$$C \cdot \mathbf{softmax}(qk^T)v \tag{12}$$

where C is the matrix given by Theorem 4.4. For multi-head attention we condition each head thereby forming

$$[C_1 \cdot \mathbf{softmax}(q_1 k_1^T)v_1, \ldots, C_h \cdot \mathbf{softmax}(q_h k_h^T)v_h]. \tag{13}$$

We call such an attention mechanism preconditioned self-attention. Note that although the theory presented in this section focuses on an upper bound of self-attention our main insight, as show in Section 5, is that this still serves as a good way to motivate the need to condition self-attention. Furthermore, the theoretical framework focused on self-attention, however in Section 5 we will show that the preconditioner matrix obtained in Theorem 4.4 works well with other forms of attention. In Section 5 we plot condition numbers of transformers.

Algorithm 1 illustrates how preconditioned attention is incorporated into the forward pass of a transformer.

Importantly, the preconditioner C is treated as a fixed transformation: it is not differentiable, carries no gradients, and therefore plays no role in the backward pass of optimization.

4.3 Computational cost

In the previous section we introduced preconditioned self-attention which takes the form

$$C \cdot \mathbf{softmax}(qk^T)v \tag{14}$$

where C is a diagonal matrix, known as a preconditioner, that is built from the row norms of the self-attention matrix $\mathbf{softmax}(qk^T)v$.

Observe that since C depends on q, k and v, and these matrices are learnable, when training a transformer using preconditioned self-attention we have to recompute C during each forward pass. This will necessarily add some computational overhead to the forward pass (see Algorithm 1). In this section, we will compute the extra floating point operations (FLOPs) for the forward pass.

Let $A = \mathbf{softmax}(qk^T)v$ denote one head of the self-attention block in one layer of a transformer and let us assume that A has size $n \times \frac{d}{h}$, where h is the total number of heads. The preconditioner matrix C given by Theorem 4.4 is computed as a diagonal matrix where the i-th diagonal element C_{ii} of C is given by the inverse of the 2-norm of the i-th row of A namely.

To compute the number of FLOPs involved in computing C through each forward pass we break down the computation.

Step 1: Compute the 2-norm of each row of A. For each row $A_{i,}$ of A, where $i = 1, 2, \ldots, n$, we have:

1. **Square each element of the row:** This requires $\frac{d}{h}$ multiplications.
2. **Sum the squared elements:** This requires $\frac{d}{h} - 1$ additions.
3. **Take the square root of the sum:** This requires 1 operation.

Thus, for a single row, the total operations are:

$$\frac{d}{h} + \left(\frac{d}{h} - 1\right) + 1 = 2\frac{d}{h}. \tag{15}$$

For n rows, the total operations to compute the 2-norms are:

$$n \cdot 2\frac{d}{h} = 2n\frac{d}{h}. \tag{16}$$

Step 2: Compute the inverse of each 2-norm. For each of the n rows, we then need to compute the reciprocal of the 2-norm:

$$C_{ii} = \frac{1}{||A_{i,}||_2}. \tag{17}$$

This requires 1 division per row, for a total of:

$$n \text{ (divisions)}. \tag{18}$$

Total operations: Summing up the operations from step 1 and step 2 we get that the total number of FLOPs is

$$\text{Total FLOPs for computing C } = n(2\frac{d}{h} + 1). \tag{19}$$

As there are h heads per layer we then get that the total number of extra FLOPs for preconditioned attention per layer is

$$\text{Total extra FLOPs per layer } = n(2d + h) \tag{20}$$

which is obtained by summing Equation (19) h times.

5 Experiments

5.1 Image Classification

In this section, we apply our theoretical insights from Section 4.2 to vision transformers on the ImageNet-1k dataset, a benchmark with over 1.2 million labeled images across 1,000 categories. We evaluate four representative architectures: ViT (Dosovitskiy et al., 2020), DeiT (Touvron et al., 2021), Swin (Liu et al., 2021), and XCiT (Ali et al., 2021). ViT-Base (ViT-B) and DeiT-Base (DeiT-B) follow the standard transformer design, with DeiT using a data-efficient training strategy. Swin-Base (Swin-B) introduces hierarchical representations with shifted window attention, while XCiT-M24 (XCiT-M) employs Local Patch Interaction and Cross-Covariance Attention. In all cases, we compare the baseline model to a variant with preconditioned attention, where each attention matrix is multiplied by the preconditioner defined in Theorem 4.4.

Results. We compared five vision transformers using their original attention mechanisms against variants with preconditioned attention. ViT-Base (Dosovitskiy et al., 2020) and DeiT-Base (Touvron et al., 2021) employ standard self-attention, while Swin-Base (Liu et al., 2021) and XCiT-Medium (Ali et al., 2021) use more specialized mechanisms. Including these latter models demonstrates that our method, though developed in the context of self-attention, extends as a drop-in replacement for modern attention designs. Full implementation details are provided in Section B.0.1. As shown in Table 1, preconditioned attention consistently improves Top-1% accuracy on ImageNet-1k.

Table 1: Comparison of vision transformers with their original attention mechanism versus one with a preconditioned attention mechanism pretrained on the ImageNet-1k dataset. We report the classification top-1% accuracy.

	Models			
	ViT-base	XciT-medium	DeiT-base	Swin-base
Original	80.2	82.6	81.6	83.4
Preconditioned	**81.4**	**83.5**	**82.7**	**84.5**

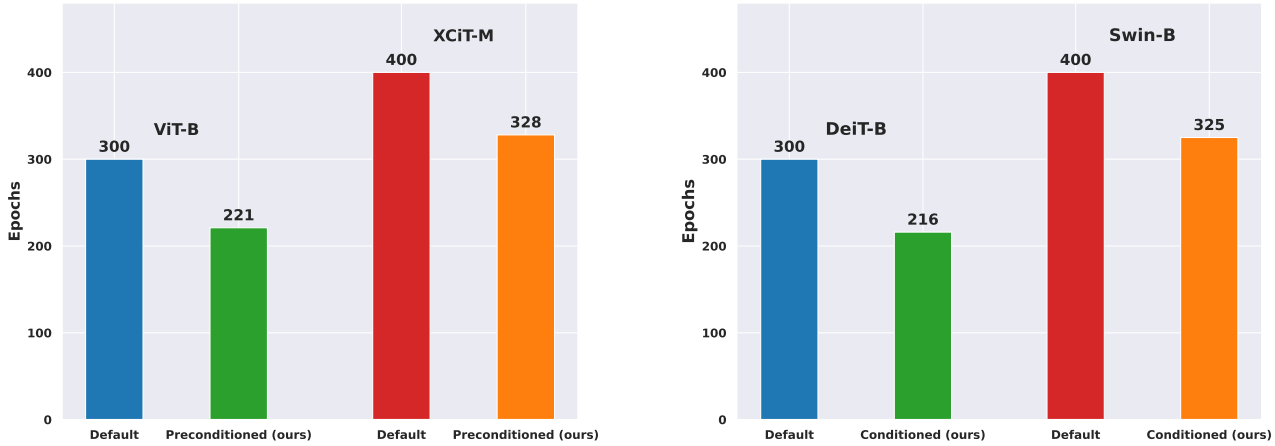

Figure 2: Total number of epochs required for each preconditioned model to reach the accuracy of the baseline model. For each ViT we see that the preconditioned model requires roughly 20-30% less epochs.

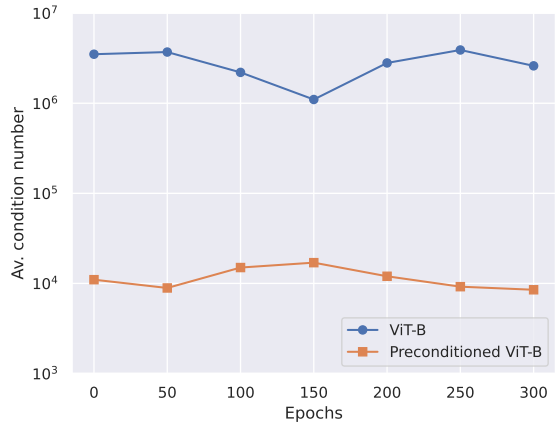

Figure 3: Average condition number of a ViT-B and a preconditioned ViT-B during training on the ImageNet-1k dataset.

Testing the theory. We evaluated the effect of preconditioning by training both a ViT-B and a preconditioned ViT-B on the ImageNet-1k dataset. During training, we computed the condition number for each attention head in every layer, averaging first across the 12 heads per layer and then across the 12 layers. The results, shown in Figure 3, indicate that the preconditioned ViT-B consistently maintains a substantially lower average condition number throughout training. Furthermore, Figure 2 compares convergence speed, showing that the preconditioned model reaches the baseline's final accuracy in roughly 20–30% fewer epochs. We note that preconditioned attention introduces additional FLOPs, as shown in Section 4.3, leading to increased training time and memory usage as reported in Table 2. However, the observed overhead remains modest, indicating that the performance gains come at only a minor computational cost.

Table 2: Training time/memory overhead for ViTs

	Train Time(hrs:mins)	**Memory** (GB)
ViT-Base	29:28	176.1
Precond. ViT-Base	30:45	178.5
DeiT-Base	26:16	173.6
Precond. DeiT-Base	27:25	175.1
Swin-Base	53:11	150.4
Precond. Swin-Base	54:24	152.6
XCiT-Med.	91:08	186.4
Precond. XCiT-Med.	92:10	188.5

6 Transfer Learning for Detection and Segmentation

In this section, we investigate the transfer learning ability of our models by evaluating them on object detection and instance segmentation. We fine-tune XCiT models pretrained on ImageNet-1k and assess their performance on the COCO 2017 benchmark (Lin et al., 2014), which contains 118K training images and 5K validation images across 80 object categories. The XCiT backbone is integrated into the Mask R-CNN framework (He et al., 2017), equipped with a Feature Pyramid Network (FPN) to capture multi-scale representations.

To interface XCiT with FPN, we adapt its columnar architecture by extracting intermediate features at multiple layers. For both XCiT-S12 and XCiT-M24, the native stride-16 outputs are converted into multi-scale features with strides $[4, 8, 16, 32]$. This is achieved through max pooling for downsampling and a transposed convolution for upsampling. Fine-tuning is carried out for 36 epochs with the AdamW optimizer, using a learning rate of 10^{-4}, weight decay of 0.05, and a batch size of 16. These experiments demonstrate that XCiT's learned representations transfer effectively to detection and segmentation tasks, underscoring the flexibility of the architecture beyond image classification.

Results. We pretrained four models on the ImageNet-1k dataset: the standard XCiT-S12 and XCiT-M24, along with their counterparts incorporating a preconditioned attention layer. The latter modifies the attention mechanism by applying the diagonal preconditioner introduced in Theorem 4.4 to the final attention output before multiplication with the values matrix. The transfer learning results are summarized in Table 3. Across both model sizes, the variants equipped with preconditioned attention consistently outperform their standard counterparts, highlighting the benefits of our proposed conditioning approach.

6.1 Long Range Sequence Modeling

Transformers often struggle with modeling long-range dependencies. The Long-Range Arena (LRA) benchmark (Tay et al., 2020) provides a standardized suite of five tasks—ListOps, text classification, retrieval, image classification, and Pathfinder, specifically designed to evaluate this capability. For detailed descriptions of these tasks, we refer the reader to Tay et al. (2020). In our experiments, we use the Nyströmformer (Xiong et al., 2021b), which approximates attention in near-linear time, and compare its standard form against a variant enhanced with preconditioned attention.

Results. We evaluated both the standard Nyströmformer and its preconditioned variant across all five tasks in the LRA benchmark suite. For each task, we strictly followed the training protocol of Xiong et al. (2021b), as detailed in their public implementation Xiong et al. (2021a). The final accuracies are reported in Table 4, where the preconditioned Nyströmformer consistently outperforms the baseline across all tasks. For details on memory usage and training time overhead, see Section B.0.3.

Testing the theory. Our first experiment examines the conditioning of the attention matrix during training on the text and ListOps tasks. We compare the standard Nyströmformer with its preconditioned variant by tracking the condition number of each attention head. As shown in Figure 4, the preconditioned Nyströmformer consistently maintains lower condition numbers (left and middle) throughout training. The right panel of Figure 4 reports the number of iterations required to match the final accuracy of the baseline model. In both tasks, the preconditioned variant converges more quickly, requiring fewer iterations to reach the same performance.

6.2 Language Modeling

Building on the theoretical insights from Section 4.2, we evaluate preconditioned attention in the Crammed BERT language model (Geiping & Goldstein, 2023) trained with masked language modeling. We pretrain two variants on The Pile (Gao et al., 2021): a standard baseline and a model with preconditioned attention following the implementation in Algorithm 1. Their performance is then assessed on the GLUE benchmark (Wang et al., 2018).

Results. Table 5 reports the GLUE benchmark performance for the default Crammed BERT and the variant with preconditioned attention. The results show that preconditioned attention consistently yields higher scores across tasks, demonstrating its effectiveness.

7 Limitations

Our work introduces a preconditioner matrix aimed at improving the conditioning of the self-attention block in transformer layers. The design is motivated by Theorem 4.4, which leverages the complexity measure in Equation (10) as an upper bound on the condition number. Although our experiments validate this measure as a useful proxy, it remains an indirect approach. A more explicit method for computing the condition number of the self-attention matrix could provide a stronger foundation for preconditioning and potentially

Table 3: Performance on COCO object detection and instance segmentation (mini-val set) using Mask R-CNN with ImageNet-1k pretrained backbones. Reported metrics include bounding box AP (AP^b), AP at IoU thresholds 0.50 and 0.75 (AP^b_{50}, AP^b_{75}), mask AP (AP^m), and mask AP at IoU thresholds 0.50 and 0.75 (AP^m_{50}, AP^m_{75}).

Model	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
XCiT-S12	44.9	66.1	48.9	40.1	63.1	42.8
Preconditioned XCiT-S12	**45.4**	**66.4**	**49.5**	**40.5**	**63.3**	**43.2**
XCiT-M24	45.7	66.8	49.6	40.8	63.6	43.3
Preconditioned XCiT-M24	**46.0**	**67.2**	**49.9**	**41.2**	**63.9**	**44.0**

Table 4: Comparison of a Nyströmformer with a preconditioned Nyströmformer on LRA benchmark. We report evaluation accuracy (%).

Model	ListOps	Text	Retrieval	Image	Pathfinder
Nyströmformer	37.1	63.8	79.8	39.9	72.9
Preconditioned Nyströmformer	**37.9**	**64.8**	**80.7**	**40.8**	**73.8**

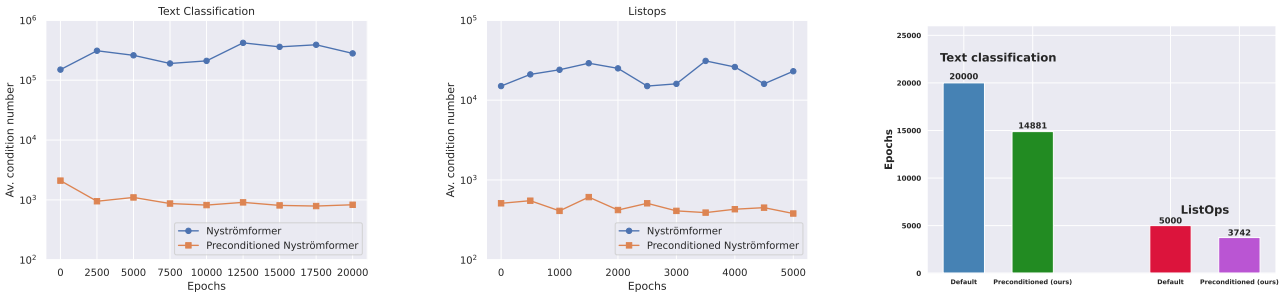

Figure 4: Conditioning analysis of the Nyströmformer on text classification and ListOps. The preconditioned variant achieves consistently lower condition numbers (left and middle) and converges faster, requiring fewer iterations to reach the baseline model's final accuracy (right).

lead to further performance gains.

8 Conclusion

We introduced a theoretical framework for analyzing the conditioning of the self-attention matrix in transformer layers and proposed a diagonal preconditioner that reduces its condition number. This yields a better-conditioned self-attention mechanism, which we termed *preconditioned self-attention*. Our empirical results across image classification, long-range sequence modeling, and language modeling demonstrate that this approach consistently improves performance. Notably, preconditioned self-attention acts as a simple drop-in replacement for existing attention mechanisms, providing broad applicability and consistent gains.

References

Naman Agarwal, Pranjal Awasthi, and Satyen Kale. A deep conditioning treatment of neural networks. In *Algorithmic Learning Theory*, pp. 249–305. PMLR, 2021.

Paul Albert, Frederic Z Zhang, Hemanth Saratchandran, Cristian Rodriguez-Opazo, Anton van den Hengel, and Ehsan Abbasnejad. Randlora: Full-rank parameter-efficient fine-tuning of large models. *arXiv preprint arXiv:2502.00987*, 2025a.

Paul Albert, Frederic Z Zhang, Hemanth Saratchandran, Anton van den Hengel, and Ehsan Abbasnejad. Towards higher effective rank in parameter-efficient fine-tuning using khatri-rao product. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1292–1302, 2025b.

Table 5: Evaluation of pretrained Crammed BERT using the default baseline and a variant with preconditioned attention (as defined in Algorithm 1). The preconditioned model consistently outperforms the baseline across GLUE tasks.

	MNLI	SST-2	STSB	RTE	QNLI	QQP	MRPC	CoLA	Avg.
Default	83.8	92.3	86.3	55.1	90.1	87.3	85.0	48.9	78.6
Preconditioned	**84.9**	**92.9**	**87.5**	**55.9**	**90.9**	**88.4**	**85.9**	**50.0**	**79.6**

Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *Advances in neural information processing systems*, 34:20014–20027, 2021.

Shin-Fang Chng, Hemanth Saratchandran, and Simon Lucey. Preconditioners for the stochastic training of neural fields. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 27222–27232, 2025.

Mingyu Ding, Bin Xiao, Noel Codella, Ping Luo, Jingdong Wang, and Lu Yuan. Davit: Dual attention vision transformers. In *European conference on computer vision*, pp. 74–92. Springer, 2022.

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. *arXiv preprint arXiv:2010.11929*, 7, 2020.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Aadi Thite, Eric Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2021.

Jonas Geiping and Tom Goldstein. Cramming: Training a language model on a single gpu in one day. In *International Conference on Machine Learning*, pp. 11117–11143. PMLR, 2023.

Heinrich W Guggenheimer, Alan S Edelman, and Charles R Johnson. A simple estimate of the condition number of a linear system. *The College Mathematics Journal*, 26(1):2–5, 1995.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Yiping Ji, Hemanth Saratchandran, Cameron Gordon, Zeyu Zhang, and Simon Lucey. Sine activated low-rank matrices for parameter efficient learning. *arXiv preprint arXiv:2403.19243*, 2024.

Yiping Ji, James Martens, Jianqiao Zheng, Ziqin Zhou, Peyman Moghadam, Xinyu Zhang, Hemanth Saratchandran, and Simon Lucey. Cutting the skip: Training residual-free transformers. *arXiv preprint arXiv:2510.00345*, 2025a.

Yiping Ji, Hemanth Saratchandran, Peyman Moghadam, and Simon Lucey. Always skip attention. *arXiv preprint arXiv:2505.01996*, 2025b.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59:85–116, 2022.

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.

Lachlan MacDonald, Jack Valmadre, Hemanth Saratchandran, and Simon Lucey. On skip connections and normalisation layers in deep optimisation. *Advances in Neural Information Processing Systems*, 36:14705–14724, 2023.

Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. *arXiv preprint arXiv:2202.08791*, 2022.

Hemanth Saratchandran and Simon Lucey. Enhancing transformers through conditioned embedded tokens. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4786–4795, 2025.

Hemanth Saratchandran and Simon Lucey. Spectral conditioning of attention improves transformer performance. *arXiv preprint arXiv:2603.07162*, 2026.

Hemanth Saratchandran, Jianqiao Zheng, Yiping Ji, Wenbo Zhang, and Simon Lucey. Rethinking attention: Polynomial alternatives to softmax in transformers. *arXiv preprint arXiv:2410.18613*, 2024.

Hemanth Saratchandran, Damien Teney, and Simon Lucey. Leaner transformers: More heads, less depth. *arXiv preprint arXiv:2505.20802*, 2025a.

Hemanth Saratchandran, Thomas X Wang, and Simon Lucey. Weight conditioning for smooth optimization of neural networks. In *European Conference on Computer Vision*, pp. 310–325. Springer, 2025b.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

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.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

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

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Fei Tan, Glenn Fung, Vikas Singh, Xiaodong Yuan, Sungsoo Ahn Wang, Dimitris Papailiopoulos, and Katerina Fragkiadaki. Github repository, 2021a. URL `https://github.com/mlpen/Nystromformer`.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 14138–14148, 2021b.

Li Yuan, Qibin Hou, Zihang Jiang, Jiashi Feng, and Shuicheng Yan. Volo: Vision outlooker for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):6575–6586, 2022.

Jianqiao Zheng, Xueqian Li, Hemanth Saratchandran, and Simon Lucey. Structured initialization for vision transformers. *arXiv preprint arXiv:2505.19985*, 2025.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Not Applicable]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Not Applicable]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Appendix

A Theoretical Analysis

In this section we give the proofs of Theorem 4.3 and Theorem 4.4.

We will start by giving the proof of Theorem 4.3. In order to do this we will need a lemma about the **softmax** activation function.

Lemma A.1. *Let $A \in \mathbb{R}^{n\times d}$ be an arbitrary $n \times d$ matrix. Let* **softmax**(A) *denote the matrix given by applying* **softmax** *row wise. Then we have that*

$$||\textbf{softmax}(A)||_F \leq \sqrt{n}. \tag{21}$$

Proof. For ease of notation let $B = \textbf{softmax}(A)$ which is an $n \times d$ matrix. Let B_{ij} denote the entry of B in the i-th row and jth column. We then have

$$\begin{aligned} ||B||_F^2 &= |B_{11}|^2 + \cdots + |B_{1d}|^2 && (22)\\ &+ && (23)\\ &\vdots && (24)\\ &+ && (25)\\ &|B_{n1}|^2 + \cdots + |B_{nd}|^2. && (26) \end{aligned}$$

Then observe that for each $1 \leq i \leq n$ we have that

$$\begin{aligned} |B_{i1}|^2 + \cdots + |B_{id}|^2 &\leq (|B_{i1}| + \cdots + |B_{id}|)^2 && (27)\\ &= 1 && (28) \end{aligned}$$

where the equality comes from the fact that we applied **softmax** row wise. Doing this for each of the n rows of B we get

$$||B||_F^2 \leq n \tag{29}$$

which gives the desired result. □

Proof of Theorem 4.3. We start by using Guggenheimer et al. result from Theorem 4.2 and compute the term

$$\left(\frac{||\textbf{softmax}(qk^T)v||_F}{\sqrt{k}}\right)^k. \tag{30}$$

We observe that $||\textbf{softmax}(qk^T)v||_F \leq ||\textbf{softmax}(qk^T)||_F \cdot ||v||_F$ and by Theorem A.1 we have that $||\textbf{softmax}(qk^T)||_F \leq \sqrt{n}$. This implies that

$$\left(\frac{||\textbf{softmax}(qk^T)v||_F}{\sqrt{k}}\right)^k \leq \left(\frac{\sqrt{n}||v||_F}{\sqrt{k}}\right)^k \tag{31}$$

which proves the theorem. □

We can now give the proof of Theorem 4.4. In order to do this we will need the following lemma.

Lemma A.2. *Let $a_1, \ldots, a_n$ be n non-negative numbers such that $a_1 \leq \cdots \leq a_n$ and such that*

$$a_1 + \cdots + a_n = n. \tag{32}$$

Then for any $1 \leq k \leq n$ we must have that

$$a_1 + \cdots + a_k \leq k. \tag{33}$$

Proof. Suppose that $a_1 + \cdots + a_k > k$. We then have that

$$n = a_1 + \cdots + a_n > k + a_{n+1} + \cdots + a_n \tag{34}$$

which implies

$$a_{k+1} + \cdots + a_n < n - k \tag{35}$$

However, since we are assuming $a_1 + \cdots + a_k > k$ we must have that at least one of the $a_i > 1$ for $1 \leq i \leq k$. This then gives a contradiction as each $a_{k+j} \geq a_i$ for $1 \leq i \leq k$ and $1 \leq j \leq n-k$ which then implies

$$a_{k+1} + \cdots + a_n \geq (n-k)a_i \geq n - k \tag{36}$$

which contradicts Equation (35). Therefore, we must have that $a_1 + \cdots + a_k \leq k$ which proves the result. □

Proof of Theorem 4.4. Our goal is to prove that $\mu(C \cdot \mathbf{softmax}(qk^T)v) \leq \mu(\mathbf{softmax}(qk^T)v)$ where μ is defined by Equation (10). For this proof, let $A := \mathbf{softmax}(qk^T)v$. It suffices for us to prove that

$$\frac{\mu(A)}{\mu(CA)} \geq 1. \tag{37}$$

Writing out the definitions of $\mu(A)$ and $\mu(C)$ we want to show that

$$\left(\frac{2}{\sigma_1(A)\cdots\sigma_k(A)}\left(\frac{||A||_F}{\sqrt{k}}\right)^k\right) \Bigg/ \left(\frac{2}{\sigma_1(CA)\cdots\sigma_k(CA)}\left(\frac{||CA||_F}{\sqrt{k}}\right)^k\right) \geq 1. \tag{38}$$

Observe that by since C is a diagonal $n \times n$ matrix whose i-th diagonal element is $1/||A_{i,}||_2$ we have that $||CA||_F = \sqrt{n}$. Furthermore, observe that for any positive number $\lambda > 0$ we have that

$$\mu(\lambda A) = \mu(A) \tag{39}$$

which follows from the fact that $||\lambda A||_F = \lambda ||A||_F$ and that for each $1 \leq i \leq k$ we have $\sigma_i(\lambda A) = \lambda \sigma_i(A)$. This implies we can assume $||A||_F = a \cdot \sqrt{n}$ for some fixed $a > 0$. For if not we can just scale A appropriately and work with the scaled A due to Equation (39). We then find that

$$\left(\frac{2}{\sigma_1(A)\cdots\sigma_k(A)}\left(\frac{||A||_F}{\sqrt{k}}\right)^k\right) \Bigg/ \left(\frac{2}{\sigma_1(CA)\cdots\sigma_k(CA)}\left(\frac{||CA||_F}{\sqrt{k}}\right)^k\right) = \frac{\sigma_1(CA)\cdots\sigma_k(CA)\cdot a^k}{\sigma_1(A)\cdots\sigma_k(A)}. \tag{40}$$

We then use the fact that for any matrices we have

$$\sigma_i(CA) \geq \sigma_{\min}(C)\sigma_i(A) \tag{41}$$

where $\sigma_{\min}(C)$ denotes the smallest singular value of C. Note that by definition C must have rank n as it is a $n \times n$ diagonal matrix built out of the inverses of the 2-norms of A and hence no diagonal entry of C can be zero.

We then obtain

$$\frac{\sigma_1(CA)\cdots\sigma_k(CA)\cdot a^k}{\sigma_1(A)\cdots\sigma_k(A)} \geq \sigma_{\min}(C)^k \cdot a^k. \tag{42}$$

It now follows that by choosing $a > 0$, in particular choosing $a \geq \frac{1}{\sigma_{\min}(C)}$, large enough we can obtain

$$\sigma_{\min}(C)^k \cdot a^k \geq 1 \tag{43}$$

which completes the proof.

□

B Experiments

Each experiment in Section 5 was repeated five times with different random seeds, and the reported results correspond to the mean across these runs.

B.0.1 Image Classification

Hardware and Implementation. Each image classification experiment was run five time with five different random seeds and the mean results given in Section 5.1. The hyperparameters and training methodology for each ViT followed the original papers: ViT (Dosovitskiy et al., 2020), DeiT (Touvron et al., 2021), Swin (Liu et al., 2021), and XCiT (Ali et al., 2021). All the experiments were carried out on Nvidia A100 GPUs. The implementation of the ViTs were all done using the Timm code base Wightman (2019)

B.0.2 Object detection and instance segmentation

Hardware and Implementation. All the experiments were carried out on Nvidia A100 GPUs. The implementation followed He et al. (2017) using the GitHub: `https://github.com/matterport/Mask_RCNN`

B.0.3 Nyströmformer on LRA benchmark

Hardware and Implementation: All the experiments were carried out on Nvidia A100 GPUs following the implementation and hyperparameter settings given in Xiong et al. (2021a). Table 6 shows the memory overhead for using a preconditioned Nyströmformer and Table 7 shows the total training time on the whole LRA benchmark.

Table 6: Memory used for training Nyströmformer and preconditioned Nyströmformer on LRA benchmark.

	Listops (GB)	Text Class. (GB)	Retrieval (GB)	Image Class. (GB)	Pathfinder (GB)
Original	5.6	3.6	13.8	9.6	9.6
Preconditioned	5.8	3.6	15.8	9.6	9.6

Table 7: Total training time for Nyströmformer and preconditioned Nyströmformer on LRA benchmark.

	Total train time on LRA (hrs:mins)
Nyströmformer	8:30
preconditioned Nyströmformer	8:37

B.1 Language modeling

Hardware and Implementation: All the experiments were carried out on a Nvidia A6000 GPU following the implementation and hyperparameter settings given in Geiping & Goldstein (2023).