Enhancing Transformers Through Conditioned Embedded Tokens

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

Simon Lucey
Australian Institute for Machine Learning
Adelaide University
simon.lucey@adelaide.edu.au

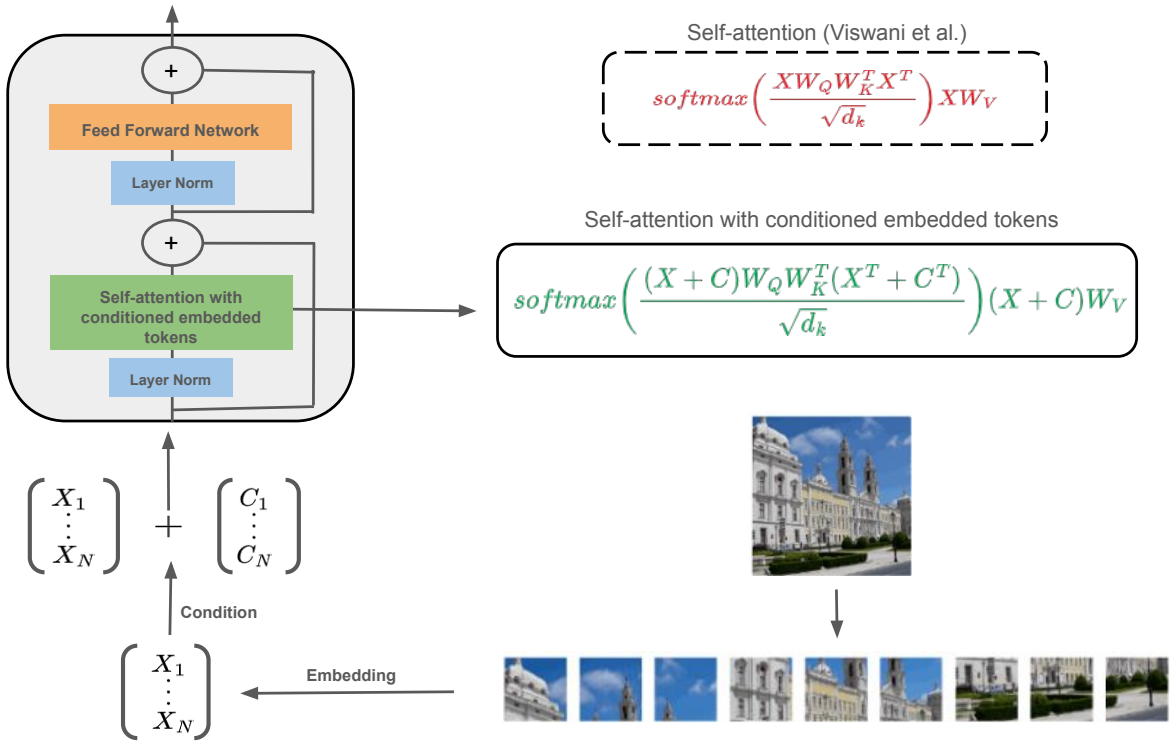

Figure 1. **Schematic representation of conditioned embeddings for a vision transformer:** An image is divided into N patches, with each patch then embedded as a high-dimensional vector $X_i \in \mathbb{R}^d$ for $1 \leq i \leq N$. These vectors are then concatenated to form the embedded token matrix $X = [X_1 \cdots X_N]^T$. To improve the conditioning of the self-attention mechanism, a correction term $C = [C_1 \cdots C_N]^T$ is added to X, reducing its condition number. The modified matrix is then fed into the first layer of the transformer (positional encoding not shown). Its effect on the self-attention equation is illustrated in the green equation (for simplicity, layer normalization has been omitted).

Abstract

Transformers have transformed modern machine learning, driving breakthroughs in computer vision, natural language processing, and robotics. At the core of their success lies the attention mechanism, which enables the modeling of global dependencies among input tokens. However, we reveal that the attention block in transformers suffers from inherent ill-conditioning, which hampers gradient-based optimization and leads to inefficient training. To address this, we develop a theoretical framework that establishes a direct relationship between the conditioning of the attention block and that of the embedded tokenized data. Building on this insight, we introduce conditioned embedded tokens, a method that systematically modifies the embedded tokens to improve the conditioning of the attention mechanism. Our analysis demonstrates that this approach significantly mitigates ill-conditioning, leading to more stable and efficient training. We validate our methodology across various transformer architectures, achieving consistent improvements in image

classification, object detection, instance segmentation, and natural language processing, highlighting its broad applicability and effectiveness.

1. Introduction

The transformer architecture [23] has become a cornerstone of modern machine learning, driving breakthroughs across various domains including computer vision [3, 5, 15, 22], natural language processing [23, 27, 28], and robotics [16, 18]. Its success is largely attributed to its ability to model complex relationships in data without relying on traditional recurrent or convolutional structures. At the heart of the transformer is the self-attention mechanism, which dynamically assigns importance to different parts of the input, capturing both local and global dependencies. Unlike traditional architectures that process inputs in a fixed order, transformers utilize self-attention to integrate global context at each layer, making them highly effective for tasks that require contextual understanding.

In general, transformers process input data by first converting it into discrete tokens, which are then mapped to a high-dimensional vector space through an embedding layer. For text, these tokens represent words or sub-words, while for images, image patches are treated as tokens. The resulting embedded tokens form a matrix that is passed into the first transformer layer for self-attention computations. This process allows the model to effectively learn relationships between tokens in a high-dimensional space.

In this paper, we explore the conditioning of the self-attention matrix in transformer architectures—an essential yet often overlooked factor influencing optimization dynamics. The conditioning of a matrix is measured by its condition number, defined as the ratio of its largest singular value to its smallest. A high condition number indicates ill-conditioning, which is a well-known challenge for the convergence of gradient-based optimization methods [17].

Previous work has addressed conditioning in feedforward neural networks. In [20], a methodology was introduced to condition network weights, leading to improved optimization. Similarly, [14] examined the condition of the neural tangent kernel (NTK), proposing a framework for reducing its condition number and demonstrating enhanced convergence for gradient descent. While these studies have focused on conditioning in feedforward architectures, its role in self-attention mechanisms remains largely unexplored. This gap motivates our investigation into how conditioning affects optimization in transformers.

We develop a theoretical framework to analyze the condition number of the self-attention matrix within a transformer's attention layer. Our analysis establishes a direct relationship between the condition number of the first-layer self-attention matrix and that of the embedded tokens. Leveraging this insight, we introduce conditioned embedded tokens, a method that applies a structured correction term to the embedded token matrix, effectively reducing its condition number and, in turn, improving the conditioning of the self-attention matrix. While a full theoretical proof linking this improvement to better convergence in gradient-based optimization would require analyzing the NTK of a transformer—a notoriously difficult task—we provide strong empirical evidence that conditioned embedded tokens enhances performance across a range of transformer applications.

Figure 1 illustrates how conditioned embedded tokens are integrated into a vision transformer. An image is first divided into tokens via patches, which are vectorized and passed through a learnable embedding layer, producing embedded tokens in a high-dimensional vector space. These tokens form the matrix $X = [X_1 \cdots X_N]^T$. A correction term $C = [C_1 \cdots C_N]^T$ is added to X, reducing its condition number before the modified embedding matrix $X+C$ is fed into the first transformer layer (positional encoding not shown). This correction not only lowers the condition number of the self-attention matrix in the first layer but also has a cascading effect, improving conditioning in subsequent layers. In general, positional encodings are added to the tokens before they are fed into the first transformer layer; however, they are omitted in Figure 1 for simplicity. Our theory will show that the optimal correction term can be derived from the singular value decomposition (SVD) of X.

We validate our framework across various applications, including image classification, object detection, instance segmentation, and natural language processing. A key advantage of conditioned embedded tokens is their flexibility—they can be used with a wide range of attention mechanisms, making them compatible with modern transformer architectures. As we demonstrate, our methodology can serve as a drop-in enhancement for advanced attention mechanisms proposed in recent works [2, 4, 15, 22, 25, 26], offering a simple yet effective improvement to existing models. Our main contributions include:

1. A theoretical framework introducing conditioned embedded tokens for improving the conditioning of the self-attention matrix within the first transformer layer yielding a more stable transformer architecture.
2. An empirical evaluation of conditioned embedded tokens across various transformer architectures and applications, including image classification, object detection, instance segmentation, and natural language processing, showing in each case its superior performance.

2. Related Work

Attention: Various strategies have been introduced to enhance the efficiency and effectiveness of transformers, particularly by addressing their computational overhead and re-

thinking attention mechanisms. The Data-Efficient Image Transformer (DeiT) [22] improves training efficiency in vision transformers by leveraging distillation tokens, allowing for competitive performance without requiring massive datasets. The Cross-Covariance Image Transformer (XCiT) [2] redefines attention by operating on cross-covariances of spatial features, enabling more efficient spatial interactions while reducing computational demands. The Nyströmformer [25], approximates standard self-attention using the Nyström method, reducing quadratic complexity to near-linear. This technique has proven particularly effective for long-range sequence modeling. In this paper, we demonstrate that our conditioning framework can be integrated as a drop-in replacement into these advanced attention mechanisms, yielding superior performance across multiple architectures.

Conditioning: In [20], the authors investigate the condition number of weight matrices in feedforward neural networks and demonstrate that lower condition numbers lead to improved accuracy across various applications. They propose a preconditioner matrix that multiplies each layer's weight matrix, effectively reducing its condition number and enhancing the training of dense weights. In [14], neural network optimization is explored through the lens of the neural tangent kernel. The authors introduce the PL* condition, a variant of the Polyak-Lojasiewicz condition, and show that this approach improves the conditioning of the neural tangent kernel, which, in the infinite-width limit, governs the dynamics of gradient descent [10]. Finally, [1] demonstrates that increasing the depth of feedforward neural networks improves their conditioning leading to more efficient optimization. See [11, 12, 19, 20] for further results.

3. Notation

In this section, we formally define the transformer architecture by describing the structure of a transformer layer, along with establishing notation for various mathematical elements that will be referenced in subsequent sections. For further foundational details on transformers, readers may consult Vaswani et al. [23].

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. In general, layer normalization is added however for simplicity we omit layer normalization for this discussion.

To begin with input data x is tokenized into N tokens yielding a tokenized representation $(x_1, \cdots, x_N)$ where $x_i \in \mathbb{R}^{t\times 1}$. Each x_i is then mapped to a high dimensional space $\mathbb{R}^{d\times 1}$ using a learnable embedding layer $E \in \mathbb{R}^{d\times t}$ so that

$$Ex_i \in \mathbb{R}^{d\times 1}. \tag{3}$$

The matrix $[Ex_1 \cdots Ex_n]^T \in \mathbb{R}^{N\times d}$ represents the embedded tokens associated to the input data. To capture the order of the tokens, a positional encoding matrix $P \in \mathbb{R}^{N\times d}$ is added forming $X = [Ex_1 \cdots Ex_n]^T + P \in \mathbb{R}^{N\times d}$. The matrix X is then input into the first layer of the transformer. For the theoretical Sec. 4 we will often use the notation X to denote embedded tokens often leaving out explicit mention of the positional encoding part for ease of notation.

The key component of the transformer is self-attention. This is composed of three learnable matrices, a query (W_Q), key (W_K), and value (W_V), where $W_Q, W_K \in \mathbb{R}^{D\times d}$ and $W_V \in \mathbb{R}^{D\times d}$. The output of the attention head $\mathbf{A}(X)$ is then given by

$$\mathbf{A}(X) = \mathbf{softmax}(XW_QW_k^TX^T)XW_V, \tag{4}$$

where $\mathbf{softmax}$ is the softmax activation that acts rowwise on the matrix $XW_QW_k^TX^T$. Note that then $A(X) \in \mathbb{R}^{N\times d}$. In the case no activation is used we have the definition of linear self-attention which we will denote by

$$\mathbf{LA}(X) := XW_QW_k^TX^TXW_V. \tag{5}$$

In general, multiple attention heads $\mathbf{A}_i$ for $1 \le i \le h$ 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,

$$[\mathbf{A}_1, \cdots, \mathbf{A}_h], \tag{6}$$

which is subsequently fed into a feedforward layer. Similary, a multi-head linear attention can be defined in the same way. The full transformer architecture is obtained by sequentially stacking several such transformer layers, as defined in Eq. (2).

4. Theoretical Framework

4.1. Conditioning of self-attention

In this section, we analyze the conditioning of the self-attention matrix in a transformer. Our objective is to demonstrate that the condition number of the self-attention matrix in the first layer is influenced by the condition number of the embedded tokens. This insight motivates our key finding: reducing the condition number of the embedded tokens leads to a better-conditioned self-attention matrix in the first layer, ultimately improving the overall conditioning of the transformer. In this section, we omit positional encoding and layer normalization to keep the core theoretical insight clear and accessible. In Sec. 5, we will show our insights go through for a variety of practical transformers that use positional encoding and layer normalization.

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

$$\kappa(A) = \frac{\sigma_1}{\sigma_k} \tag{7}$$

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 $\sigma_1 \geq \sigma_k$.

We have the following estimate for the condition number of linear self-attention and softmax self-attention.

Proposition 4.2. *Let X denote an input for an attention layer as defined in Sec. 3. Assume $\mathbf{LA}(X)$ and $\mathbf{A}(X)$ have full rank. We then have*

$$\kappa(\mathbf{LA}(X)) \leq \kappa(W_Q)\kappa(W_K)\kappa(W_V)\kappa(X)^3 \tag{8}$$

$$\kappa(\mathbf{A}(X)) \leq \kappa(\mathbf{softmax}(XW_QW_K^TX^T))\kappa(X)\kappa(W_V). \tag{9}$$

The proof of Proposition 4.2 is given in Sec. A of the Supp. material.

Remark 4.3. The statement of Proposition 4.2 assumes the attention matrices are full rank. Since the layers of a transformer are randomly initialized we have that with probability 1 the attention matrices will be full rank at initialization.

We point out that although Eq. (8) and Eq. (9) only provide an upper bound for the condition number, we found that the this upper bound provides a useful measure for estimating the condition number of the self-attention matrix. Therefore, for the self-attention matrices $\mathbf{A}(X)$ and $\mathbf{LA}(X)$ we make the following definition

$$\mu(\mathbf{LA}(X)) := \kappa(W_Q)\kappa(W_K)\kappa(W_V)\kappa(X)^3. \tag{10}$$

$$\mu(\mathbf{A}(X)) := \kappa(\mathbf{softmax}(XW_QW_k^TX^T))\kappa(X)\kappa(W_V). \tag{11}$$

4.2. Conditioned embedded tokens

In this section, we present our main theorem, which provides a method for reducing the condition number of embedded tokens derived from a dataset. This reduction subsequently decreases the measure $\mu(\mathbf{A}(X))$, a key metric for assessing the condition number of the self-attention matrix in the first transformer layer.

Theorem 4.4. *Let $(x_1, \cdots, x_N)$ denote N tokens associated to an input dataset with $x_i \in \mathbb{R}^{t\times 1}$. Let $E \in \mathbb{R}^{d\times t}$ denote an embedding matrix so that $Ex_i \in \mathbb{R}^{d\times 1}$. Let $X = [Ex_1 \cdots Ex_N]^T \in \mathbb{R}^{N\times d}$ denote the matrix of embedded tokens. Assume that $\kappa(X) > 2$. Then there exists a matrix $C \in \mathbb{R}^{N\times d}$ such that*

$$\kappa(X + C) \leq 2 \tag{12}$$

and hence that $\kappa(X + C) < \kappa(X)$.

In particular, Theorem 4.4 shows that if the condition number of the embedded tokens X is extremely large. Adding a correction term given by C will bring it down significantly. As we shall see in the experiments Sec. 5 the condition number $\kappa(X)$ is empirically extremely large. The proof of Theorem 4.4 is given in Sec. A of the Supp. material.

Theorem 4.5. *Let X denote a collection of embedded tokens, as defined in the statement of Theorem 4.4, and let C denote the matrix given by Theorem 4.4. We then have*

$$\mu(\mathbf{LA}(X + C)) \leq \mu(\mathbf{LA}(X)). \tag{13}$$

Assume that

$$\kappa(\mathbf{softmax}((X + C)W_QW_K^T(X^T + C^T))) \leq \kappa(\mathbf{softmax}(XW_QW_K^TX^T)) \tag{14}$$

then

$$\mu(\mathbf{A}(X + C)) \leq \mu(\mathbf{A}(X)) \tag{15}$$

where $\mu(\mathbf{LA}(X))$ is defined by Eq. (10) and $\mu(\mathbf{A}(X))$ by Eq. (11).

The proof of Theorem 4.5 is given in Sec. A of the Supp. material. A natural question arises: can self-attention be conditioned directly through queries, keys, and values, independent of the embedded tokens? Unfortunately, our approach of introducing a correction term does not extend to these components. This is because our correction modifies the singular values of the embedded tokens, whereas altering the singular values of queries, keys, or values could disrupt how self-attention captures spatial relationships between tokens.

Remark 4.6. We note that in the case of self-attention with a softmax activation Theorem 4.5 requires an assumption on the condition number of the non-linear self-attention probability matrix term $\mathbf{softmax}((X+C)W_QW_K^T(X^T+C^T))$. Although this may be seen as a limitation of the theory we will see in Sec. 5 that this framework still provides a useful methodology to obtain good performance on a variety of transformer applications.

Remark 4.7. Theorem 4.5 demonstrates that conditioning embedded tokens can improve the condition number of the self-attention matrix in the first layer. However, it does not address how this affects the condition number in subsequent layers. In Sec. 5, we will show that this effect propagates through the network, leading to a reduced condition number in deeper self-attention layers.

Remark 4.8. Our theory, as presented in Theorem 4.4 and Theorem 4.5, demonstrates that conditioning embedded tokens reduces the condition number of the self-attention matrix in the first layer. However, the framework does not

	Models				
	ViT-base	DeiT-base	Swin-base	XciT-medium	DaViT-base
Original embedded tokens	80.3	81.6	83.1	82.2	83.6
Conditioned embedded tokens	**81.3**	**82.5**	**83.9**	**82.9**	**84.6**

Table 1. Comparison of vision transformers with their original embedded tokens verse one with a conditioned conditioned embedded tokens pre-trained on the ImageNet-1k dataset. We report the classification top-1% accuracy. In each case we see the transformers employing conditioned embedded tokens (ours) outperforms the original ones.

theoretically establish how this improvement translates to better optimization and, consequently, higher accuracy in applications. Nonetheless, in Sec. 5, we provide empirical evidence showing that our approach consistently enhances accuracy across various applications, highlighting its practical effectiveness.

5. Experiments

In this section, we evaluate our insight from Sec. 4 on a variety of transformer applications. For a detailed analysis of how we implemented conditioned embedded tokens see Sec. A in the Supp. material.

5.1. Image Classification

In this section, we apply our theoretical insights from Sec. 4 to vision transformers on an image classification task. We will train all vision transformers from scratch on the ImageNet-1k dataset. We will be testing on the following vision transformers. In each case we will compare the original architecture with one that uses conditioned embedded tokens.

ViT [5]: ViT is a pioneering architecture that applies transformer-based processing to images by treating them as sequences of non-overlapping patches. In our study, we utilize ViT-Base (ViT-B), which is configured with a patch size of 16, an embedding dimension of 768, 12 attention heads, and 12 layers. The model employs a standard self-attention mechanism to capture dependencies across patches.

DeiT [22]: DeiT builds upon ViT but is optimized for data efficiency. Unlike standard ViT, it employs a data-starvation training strategy [22], enabling faster convergence while maintaining strong performance. In our setup, we use DeiT-Base (DeiT-B) with a patch size of 16, an embedding dimension of 768, 12 attention heads, and 12 layers. Like ViT, DeiT utilizes a self-attention mechanism for feature extraction.

Swin Transformer [15]: Swin Transformer introduces a hierarchical architecture and a novel shifted window-based self-attention mechanism, significantly improving efficiency and effectiveness in vision tasks. In our experiments, we adopt the Swin-Base (Swin-B) variant, which features 128 channels in the hidden layers of the first stage.

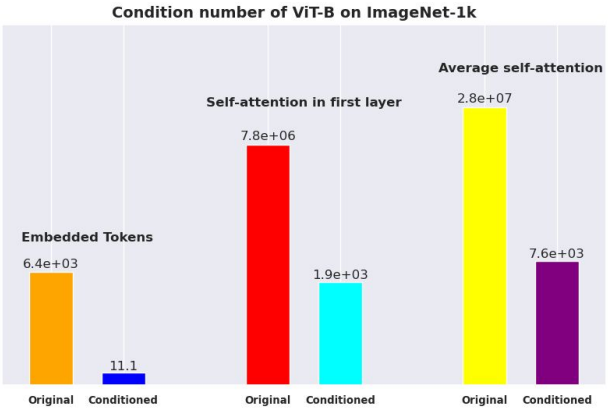

Figure 2. Condition number comparison for ViT-B on ImageNet-1k. The left, middle, and right bars show the condition number of embedded tokens, first-layer self-attention (averaged across heads), and self-attention across all layers, respectively, averaged over training epochs. Our model (conditioned) consistently achieves a significantly lower condition number than the original.

The attention operates over non-overlapping windows of size $M = 7$, while the query dimension per head is set to $d = 32$. The network follows a hierarchical structure with layers arranged as $\{2, 2, 18, 2\}$ across different stages.

XCiT [2]: XCiT diverges from standard ViT architectures by incorporating two key innovations. First, it integrates Local Patch Interaction within each block, using depth-wise 3×3 convolutions, Batch Normalization, GELU activation, and an additional depth-wise convolution to enhance local feature representation. Second, it employs Cross-Covariance Attention, where attention maps are derived from the cross-covariance matrix of the key and query projections. For our experiments, we utilize the XCiT-M24 (XCiT-M) model with a patch size of 16, an embedding dimension of 512, 8 attention heads, and 24 layers.

DaViT [4]: DaViT extends traditional vision transformers by incorporating both spatial and channel attention mechanisms, enhancing feature extraction across multiple dimensions. While conventional transformers rely solely on spatial self-attention, DaViT introduces channel attention to capture dependencies across feature channels, complement-

Model	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
XCiT-S + original embedded tokens	44.9	66.1	48.9	40.1	63.1	42.8
XCiT-S + conditioned embedded tokens	**45.7**	**66.6**	**49.7**	**40.4**	**63.5**	**43.4**
XCiT-M + original embedded tokens	45.7	66.8	49.6	40.8	63.6	43.3
XCiT-M + conditioned embedded tokens	**46.2**	**67.4**	**49.8**	**41.4**	**63.8**	**44.2**

Table 2. Performance evaluation of object detection and instance segmentation on the COCO mini-val set. The backbone networks are pretrained on ImageNet-1k and integrated into the Mask R-CNN framework. The reported metrics include AP^b (Average Precision for bounding box predictions), $AP^b_{50/75}$ (Average Precision at IoU thresholds of 0.50 and 0.75 for bounding boxes), AP^m (Average Precision for mask predictions), and $AP^m_{50/75}$ (Average Precision at IoU thresholds of 0.50 and 0.75 for mask predictions). In each case we see the XCiT architecture employing conditioned embedded tokens (ours) out performs the original architecture.

ing spatial interactions. This hybrid approach allows DaViT to effectively model both local and global relationships. In our experiments, we use the DaViT-Base (DaViT-B) model, configured with a patch size of 4, embedding dimensions of $(128, 256, 512, 1024)$, attention heads of $(4, 8, 16, 32)$, and a layer distribution of $(1, 1, 9, 1)$.

Results: Table 1 presents a comparison of the top-1% test accuracy of five vision transformers trained from scratch on the ImageNet-1k dataset. Each model is evaluated using both the original embedded tokens and our conditioned embedded tokens, following the approach outlined in Sec. 4.2. While ViT-Base [5] and DeiT-Base [22] utilize standard self-attention mechanisms, Swin-Base [15], XCiT-Medium [2], and DaViT-Base [4] incorporate more advanced attention mechanisms. We include these latter models to illustrate that, although our theoretical analysis focuses on self-attention, conditioned embedded tokens can be easily integrated into modern transformers with more sophisticated architectures. As shown in the table, models using conditioned embedded tokens consistently achieve higher accuracy. Full implementation details and hyperparameter settings can be found in Sec. B.2 of the Supp. material.

Analysis: Figure 2 illustrates the condition numbers of the embedded tokens, the first-layer self-attention matrix (averaged across all heads), and the overall self-attention matrix (averaged across all layers) for the ViT-B architecture. We compare these metrics for models using the original embedded tokens and our conditioned embedded tokens. These values were computed at each training epoch and then averaged over all epochs to generate the plots. The figure clearly shows that the ViT-B architecture with conditioned embedded tokens maintains on average a significantly lower condition number throughout training.

Implementation and hardware: For full details on the implementation of each vision transformer, the training hyperparameters and the hardware used please see Sec. B.2 in the Supp. material.

5.2. Object Detection and Image Segmentation

In this section, we extend our approach from Sec. 4.2 to object detection and instance segmentation tasks. To evaluate its effectiveness in fine-tuning scenarios, we adapt a pretrained XCiT model, originally trained on ImageNet-1k, for these tasks. Our experiments leverage the COCO 2017 dataset [13], which consists of 118K training images and 5K validation images spanning 80 object categories. For model architecture, we employ XCiT as the backbone of the Mask R-CNN framework [9], incorporating a Feature Pyramid Network (FPN) to enhance multi-scale feature representation. To integrate XCiT with FPN, we modify its inherently columnar structure by extracting features from multiple layers. We do this by using an XCiT-small that has 12 layers (XCiT-S) and a XCiT-medium (XCiT-M) that has 24 layers. These features, originally computed at a fixed stride of 16, are adjusted to strides of $[4, 8, 16, 32]$ to align with the FPN hierarchy. Max pooling is used for downsampling, while a single transposed convolution layer facilitates upsampling. The model is then fine-tuned for 36 epochs using the AdamW optimizer, with a learning rate of 10^{-4}, a weight decay of 0.05, and a batch size of 16. This training strategy effectively demonstrates how XCiT's learned representations can be adapted for downstream tasks.

Results: We pretrained a total of four models on the ImageNet-1k dataset, namely an XCiT-S and XCiT-M both using the original embedded tokens following the methodology from Sec. 4.2. The results of the experiment are shown in Tab. 2. As can be seen from the table in both cases the transformer using conditioned embedded tokens out performs the original.

Implementation and hardware: For full details on the implementation, the training hyperparameters and the hardware used see Sec. B.3 in the Supp. material.

5.3. Language Models

In this section, we evaluate our insights from Sec. 4.2 on two different language models.

	MNLI	SST-2	STSB	RTE	QNLI	QQP	MRPC	CoLA	GLUE
Crammed BERT (original)	83.8	92.3	86.3	55.1	90.1	87.3	85.0	48.9	78.6
Crammed BERT (ours)	**84.2**	**92.5**	**86.5**	**55.6**	**91.1**	**87.4**	**86.3**	**53.7**	**79.7**

Table 3. Evaluation of a pre-trained Crammed BERT on the GLUE benchmark with the original embedded tokens (original) and our conditioned embedded tokens (ours). As can be seen from the table our methodology out performs the original in every task.

Crammed BERT: In the first experiment we applied our insights from Sec. 4.2 to a Crammed BERT language model [8], trained entirely from scratch using masked language modeling with conditioned embedded tokens. The model consists of 110 million parameters, with 12 transformer layers and 12 attention heads per layer. We train two versions from scratch: the original Crammed BERT [8] and a variant incorporating conditioned embedded tokens using the insight from Sec. 4.2. Both models are trained on The Pile dataset [7], a large-scale corpus designed for language model training following the pretraining regime in [8]. After pretraining, we evaluate the performance on the GLUE benchmark [24] following the evaluation methodology outlined in [8].

Results: The evaluation results are shown in Tab. 3 with each number representing the accuracy. We see from the table that in each task the model employing conditioned embedded tokens outperforms the original Crammed Bert model on every task within the GLUE benchmark.

GPT-2: We conducted a second experiment training a GPT-2 model with a decoder-only transformer architecture, consisting of 12 layers, 12 attention heads per layer, and an embedding dimension of 512. The model was trained using masked self-attention to predict the next token, leveraging the TinyStories dataset [6] for language generation. Two versions of the model were trained from scratch: one using the original embedded tokens as in [6] and another incorporating the conditioned embedded tokens from Sec. 4.2. Both models were evaluated using perplexity loss (PPL), a widely used metric for assessing language models on datasets like TinyStories. Perplexity measures the model's confidence in its predictions by computing the exponential of the average negative log-likelihood of the predicted tokens. A lower perplexity score indicates that the model assigns higher probabilities to correct tokens, leading to better fluency and coherence in text generation.

Results and Analysis: Tab. 4 presents the results of our experiment, demonstrating that the model trained with conditioned embedded tokens consistently achieves lower validation loss and lower perplexity compared to the original GPT-2 model. These results indicate that conditioning the embedded tokens improves model performance, leading to more accurate language generation. Fig. 3 plots the condition numbers of the embedded tokens, self-attention in first layer averaged over all heads and self-attention averaged over all layers. Each of these are evaluated over each training epoch and then averaged over all epochs.

	Val loss
GPT-2 original	2.41
GPT-2 conditioned (ours)	**2.36**

Table 4. GPT-2 model trained on the TinyStories dataset. We compare two models, the original model and one using conditioned embedded tokens (conditioned). We report the validation loss; in both cases, our model yields a lower loss, indicating better performance.

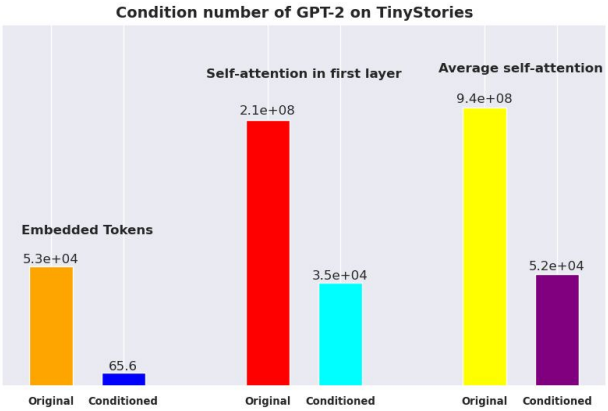

Figure 3. Condition number comparison for GPT-2 training on TinyStories. The left, middle, and right bars show the condition number of embedded tokens, first-layer self-attention (averaged across heads), and self-attention across all layers, respectively, averaged over all epochs. Our model (conditioned) consistently achieves a significantly lower condition number than the original.

Implementation and hardware: For full details on the implementation, the training hyperparameters and the hardware used please see Sec. B.4 in the Supp. material.

5.4. Nyströmformer for Long Range Sequences

We applied our methodology to the Nyströmformer [25], a transformer architecture designed for efficient long-range dependency modeling on the Long-Range Arena (LRA) benchmark [21]. We trained two Nyströmformer models:

Model	ListOps	Text	Retrieval	Image	Pathfinder
Nyströmformer (original)	37.1	63.8	79.8	39.9	72.9
Nyströmformer conditioned (ours)	**37.9**	**64.9**	**80.9**	**40.1**	**80.0**

Table 5. Comparison of a Nyströmformer using its original embedded tokens (original) with a Nyströmformer using conditioned embedded tokens (ours) on the LRA benchmark. We report the evaluation accuracy (%). As can be seen from the table our methodology yields a hihger accuracy on each task.

one utilizing conditioned embedded tokens and the original Nyströmformer for comparison. The Long-Range Arena (LRA) benchmark consists of five distinct tasks:

1. **ListOps**: A synthetic task that tests hierarchical reasoning by evaluating nested arithmetic operations.
2. **Text Classification**: Character-level text classification, requiring models to process long sequences of text.
3. **Retrieval**: A document retrieval task that measures the model's ability to identify relevant patterns within long textual inputs.
4. **Image Classification**: Modeling image data as a sequence of flattened pixel values to assess a model's capability in visual recognition.
5. **Pathfinder**: A visual reasoning challenge that requires detecting connected paths in an image, emphasizing spatial relationship modeling.

Results and Analysis: Tab. 5 shows the results of the experiment. As can be seen from the table the Nyströmformer that employed conditioned embedded tokens outperformed the original Nyströmformer from [25] in every task in the LRA benchmark. We computed the condition numbers of the original Nyströmformer and one employing conditioned embedded tokens throughout training for the embedded tokens, the self-attention within the first layer averaged over all heads and the self-attention averaged over all layers. We then took the average for each task over all training epochs and averaged over the five tasks within the LRA benchmark. The plots are shown in Fig. 4. As can be seen from the figure, the Nyströmformer employing conditioned embedded tokens has lower condition number in all cases.

Implementation and hardware: For full details on the implementation, the training hyperparameters and the hardware used please see Sec. B.5 in the Supp. material.

6. Limitations

Our work explored the conditioning of transformer attention blocks through embedded tokens. While we showed that conditioned embedded tokens reduce the first-layer self-attention condition number, their impact on optimization remains theoretically unproven. Prior studies [1, 10, 14] have linked NTK conditioning to optimization in feedforward networks, raising an important question: how does

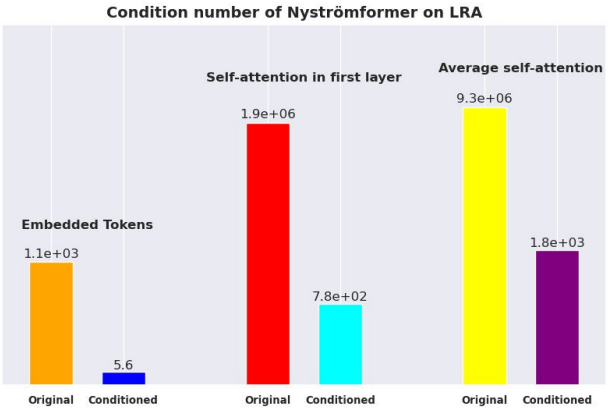

Figure 4. Condition number comparison for Nyströmformer on the LRA benchmark. The left, middle, and right bars show the condition number of embedded tokens, first-layer self-attention (averaged across heads), and self-attention across all layers, averaged over training epochs and tasks. Our model (conditioned) consistently achieves a significantly lower condition number than the original.

conditioning embedded tokens affect the NTK in transformers? Investigating this could offer deeper insights into how embedded token structure influences transformer optimization.

7. Conclusion

In this paper, we introduced a framework for analyzing the conditioning of the self-attention matrix in the first layer of a transformer, highlighting its dependence on the embedded tokens. Our analysis demonstrated that by adding a carefully designed correction term—which we refer to as conditioned embedded tokens—we could significantly reduce the condition number of self-attention in the first layer, leading to a better-conditioned attention mechanism. We empirically validated the effectiveness of conditioned embedded tokens across a diverse set of tasks, including image classification, object detection, instance segmentation, language modeling, and long-range sequence modeling. In all cases, our approach proved to be a simple, drop-in replacement for existing embedding methods, consistently improving model performance across these applications.

References

[1] Naman Agarwal, Pranjal Awasthi, and Satyen Kale. A deep conditioning treatment of neural networks. In *Algorithmic Learning Theory*, pages 249–305. PMLR, 2021. 3, 8

[2] 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. 2, 3, 5, 6

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 2

[4] Mingyu Ding, Bin Xiao, Noel Codella, Ping Luo, Jingdong Wang, and Lu Yuan. Davit: Dual attention vision transformers. In *European conference on computer vision*, pages 74–92. Springer, 2022. 2, 5, 6

[5] 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. *arXiv preprint arXiv:2010.11929*, 2020. 2, 5, 6

[6] Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023. 7

[7] 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. 7

[8] Jonas Geiping and Tom Goldstein. Cramming: Training a language model on a single gpu in one day. In *International Conference on Machine Learning*, pages 11117–11143. PMLR, 2023. 7

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 6

[10] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018. 3, 8

[11] Yiping Ji, Hemanth Saratchandran, Cameron Gordon, Zeyu Zhang, and Simon Lucey. Efficient learning with sine-activated low-rank matrices. *arXiv preprint arXiv:2403.19243*, 2024. 3

[12] Yiping Ji, Hemanth Saratchandran, Peyman Moghaddam, and Simon Lucey. Always skip attention. *arXiv preprint arXiv:2505.01996*, 2025. 3

[13] 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*, pages 740–755. Springer, 2014. 6

[14] 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. 2, 3, 8

[15] 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*, pages 10012–10022, 2021. 2, 5, 6

[16] Abhisek Maiti, Sander Oude Elberink, and George Vosselman. Transfusion: Multi-modal fusion network for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6536–6546, 2023. 2

[17] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999. 2

[18] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control. *arXiv preprint arXiv:2001.03093*, 2, 2020. 2

[19] 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. 3

[20] Hemanth Saratchandran, Thomas X Wang, and Simon Lucey. Weight conditioning for smooth optimization of neural networks. In *European Conference on Computer Vision*, pages 310–325. Springer, 2025. 2, 3

[21] 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. 7

[22] 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*, pages 10347–10357. PMLR, 2021. 2, 3, 5, 6

[23] 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. 2, 3

[24] 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. 7

[25] 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*, pages 14138–14148, 2021. 2, 3, 7, 8

[26] 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. 2

[27] Q Zhen, W Sun, H Deng, D Li, Y Wei, B Lv, J Yan, L Kong, and Y Zhong. cosformer: rethinking softmax in attention.

In *International Conference on Learning Representations*, 2022. 2

[28] Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. A robustly optimized bert pre-training approach with post-training. In *Proceedings of the 20th chinese national conference on computational linguistics*, pages 1218–1227, 2021. 2