Unveiling the Hidden Structure of Self-Attention via Kernel Principal Component Analysis

Rachel S.Y. Teo
Department of Mathematics
National University of Singapore
rachel.tsy@u.nus.edu

Tan M. Nguyen
Department of Mathematics
National University of Singapore
tanmn@nus.edu.sg

Abstract

The remarkable success of transformers in sequence modeling tasks, spanning various applications in natural language processing and computer vision, is attributed to the critical role of self-attention. Similar to the development of most deep learning models, the construction of these attention mechanisms relies on heuristics and experience. In our work, we derive self-attention from kernel principal component analysis (kernel PCA) and show that self-attention projects its query vectors onto the principal component axes of its key matrix in a feature space. We then formulate the exact formula for the value matrix in self-attention, theoretically and empirically demonstrating that this value matrix captures the eigenvectors of the Gram matrix of the key vectors in self-attention. Leveraging our kernel PCA framework, we propose Attention with Robust Principal Components (RPC-Attention), a novel class of robust attention that is resilient to data contamination. We empirically demonstrate the advantages of RPC-Attention over softmax attention on the ImageNet-1K object classification, WikiText-103 language modeling, and ADE20K image segmentation task. The code is publicly available at https://github.com/rachtsy/KPCA_code.

1 Introduction

Transformers [83] have emerged as the preeminent model for tackling a myriad of challenging problems in natural language processing [4, 20, 1, 66, 9, 39, 17], computer vision [22, 78, 43, 63, 38, 24], reinforcement learning [13, 35, 94, 40], and other applications [93, 68, 36, 88]. The effectiveness of transformers is rooted in their ability to learn from unlabeled data and take advantage of pre-trained models for downstream tasks that involve diverse data modalities with limited supervision [64, 65, 90, 97, 67]. At the core of the transformer's success lies the self-attention mechanism, which serves as the fundamental building block of a transformer model. This mechanism enables each token in a sequence to aggregate information from other tokens by computing a weighted average based on similarity scores between their feature representations. Facilitating dynamic interactions among tokens, this attention mechanism allows tokens to selectively attend to others, thereby attaining a contextual representation [5, 56, 60, 42]. The flexibility in capturing diverse syntactic and semantic relationships is an important factor contributing to the success of transformers [77, 85, 33, 84, 53].

Self-Attention. For a given input sequence $\boldsymbol{X} := [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D_x}$ of N feature vectors, self-attention transforms $\boldsymbol{X}$ into the output sequence $\boldsymbol{H}$ in the following two steps:

Step 1: The input sequence $\boldsymbol{X}$ is projected into the query matrix $\boldsymbol{Q}$, the key matrix $\boldsymbol{K}$, and the value matrix $\boldsymbol{V}$ via three linear transformations

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q^\top; \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K^\top; \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V^\top, \tag{1}$$

Please correspond to: tanmn@nus.edu.sg

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-3215

where $\boldsymbol{W}_Q, \boldsymbol{W}_K \in \mathbb{R}^{D \times D_x}$, and $\boldsymbol{W}_V \in \mathbb{R}^{D_v \times D_x}$ are the weight matrices. We denote $\boldsymbol{Q} := [\boldsymbol{q}_1, \cdots, \boldsymbol{q}_N]^\top$, $\boldsymbol{K} := [\boldsymbol{k}_1, \cdots, \boldsymbol{k}_N]^\top$, and $\boldsymbol{V} := [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N]^\top$, where the vectors $\boldsymbol{q}_i, \boldsymbol{k}_i, \boldsymbol{v}_i$ for $i = 1, \cdots, N$ are the query, key, and value vectors, respectively.

Step 2: The output sequence $\boldsymbol{H} := [\boldsymbol{h}_1, \cdots, \boldsymbol{h}_N]^\top$ is then computed as follows

$$\boldsymbol{H} = \mathrm{softmax}\Big(\boldsymbol{Q}\boldsymbol{K}^\top / \sqrt{D}\Big)\boldsymbol{V} := \boldsymbol{A}\boldsymbol{V}, \tag{2}$$

where the softmax function is applied to each row of the matrix $\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{D}$. The matrix $\boldsymbol{A} := \mathrm{softmax}\Big(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D}}\Big) \in \mathbb{R}^{N \times N}$ and its component a_{ij} for $i,\ j = 1, \cdots, N$ are called the attention matrix and attention scores, respectively. For each query vector $\boldsymbol{q}_i$ for $i = 1, \cdots, N$, an equivalent form of Eqn. (2) to compute the output vector $\boldsymbol{h}_i$ is given by

$$\boldsymbol{h}_i = \sum_{j=1}^{N} \mathrm{softmax}\Big(\boldsymbol{q}_i^\top \boldsymbol{k}_j / \sqrt{D}\Big)\boldsymbol{v}_j. \tag{3}$$

The self-attention computed by Eqn. (2) and (3) is called the scaled dot-product or softmax attention. In our paper, we call a transformer that uses this attention the softmax transformer. The structure that the attention matrix $\boldsymbol{A}$ learns from training determines the ability of the self-attention to capture contextual representations for each token.

Despite their impressive achievements, the development of most attention layers rely on intuitions and heuristic approaches. The quest for a systematic and principled framework for studying and synthesizing attention layers has remained challenging.

Contribution. We study and analyze self-attention in transformers from the perspective of kernel principal component analysis (kernel PCA). In particular, we develop a novel connection between self-attention and kernel PCA, showing that *self-attention projects its query vectors $\boldsymbol{q}_i$, $i = 1, \ldots, N$, onto principal component axes of the key matrix $\boldsymbol{K}$ in a feature space*. We then inspect the structure of the value matrix $\boldsymbol{V}$ of self-attention suggested by our kernel PCA framework, validating $\boldsymbol{V}$ *captures the eigenvectors of the Gram matrix of the key vectors $\boldsymbol{k}_j$,$j = 1, \ldots, N$*. Using our framework, we then propose a new class of robust attention, namely the Attention with Robust Principal Components (RPC-Attention). Our contribution is three-fold.

1. We derive self-attention from kernel PCA, showing that the attention outputs are projections of the query vectors onto the principal components axes of the key matrix $\boldsymbol{K}$ in a feature space.
2. We discover and validate that the value matrix $\boldsymbol{V}$ in self-attention captures the eigenvectors of the Gram matrix of the key vectors $\boldsymbol{k}_j$,$j = 1, \ldots, N$.
3. We develop the Attention with Robust Principal Components (RPC-Attention), a new attention mechanism that is resilient to data contamination, using our kernel PCA framework.

We empirically demonstrate the benefits of RPC-Attention on the ImageNet-1K object classification, ADE20K image segmentation, and large scale WikiText-103 language modeling tasks. We further illustrate the robustness of RPC-Attention through our evaluations on popular, standard robustness benchmarks, as well as various white and black box adversarial attacks on ImageNet-1K images, 15 different types of corruptions on the ADE20K dataset, and word swap attack on WikiText-103.

2 Principal Component Analysis of Attention

In this section, we will derive attention from kernel PCA. Suppose we are given a dataset $M = \{\boldsymbol{k}_1, \ldots, \boldsymbol{k}_N\} \subset \mathbb{R}^D$. Here, $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_N$ are attention keys in self-attention. As in kernel PCA, we first project these data points into a feature space using a feature map $\boldsymbol{\varphi}(\boldsymbol{x}) := \boldsymbol{\phi}(\boldsymbol{x})/g(\boldsymbol{x})$, where $\boldsymbol{\phi}(\cdot)$ is a nonlinear transformation from $\mathbb{R}^D$ to $\mathbb{R}^{D'}$, and $g(\cdot)$ is a vector-scalar function that computes a scaling factor for $\boldsymbol{\phi}(\boldsymbol{x})$. We center the projected data as follows:

$$\tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_j) = \boldsymbol{\varphi}(\boldsymbol{k}_j) - \frac{1}{N}\sum_{j'=1}^{N} \boldsymbol{\varphi}(\boldsymbol{k}_{j'}). \tag{4}$$

https://doi.org/10.52202/079017-3215

Letting $\boldsymbol{C}$ be the covariance matrix of the centered data in feature space, its eigenvector expansion is

$$\boldsymbol{C} = \frac{1}{N}\sum_{j=1}^{N} \tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_j)\tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_j)^\top; \quad \boldsymbol{C}\boldsymbol{u}_d = \lambda_d \boldsymbol{u}_d, \ \ d = 1, \dots, D_v. \tag{5}$$

Plugging $\boldsymbol{C}$ into (5), we obtain

$$\frac{1}{N}\sum_{j=1}^{N} \tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_j)\{\tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_j)^\top \boldsymbol{u}_d\} = \lambda_d \boldsymbol{u}_d. \tag{6}$$

Thus, provided that $\lambda_d > 0$, the eigenvector $\boldsymbol{u}_d$ is given by a linear combination of the $\tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_j)$ and with $a_{dj} = \frac{1}{N\lambda_d}\tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_j)^\top \boldsymbol{u}_d$, can be written as

$$\boldsymbol{u}_d = \sum_{j=1}^{N} a_{dj}\tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_j). \tag{7}$$

2.1 Deriving Attention from Kernel PCA

In order to derive self-attention from kernel PCA, we consider the query vector $\boldsymbol{q}_i$ in self-attention as a new data point. The projection $\boldsymbol{h}_i$ of a new test point $\boldsymbol{q}_i$ onto the principal components $\boldsymbol{u}_d$ in Eqn. (7), for $d = 1, \dots, D_v$, is given by

$$\begin{aligned}
\boldsymbol{h}_i(d) = \boldsymbol{\varphi}(\boldsymbol{q}_i)^\top \boldsymbol{u}_d &= \sum_{j=1}^{N} a_{dj}\boldsymbol{\varphi}(\boldsymbol{q}_i)^\top \tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_j) = \sum_{j=1}^{N} a_{dj}\boldsymbol{\varphi}(\boldsymbol{q}_i)^\top \Big(\boldsymbol{\varphi}(\boldsymbol{k}_j) - \frac{1}{N}\sum_{j'=1}^{N}\boldsymbol{\varphi}(\boldsymbol{k}_{j'})\Big) \\
&= \sum_{j=1}^{N} \frac{k(\boldsymbol{q}_i, \boldsymbol{k}_j)}{g(\boldsymbol{q}_i)} \frac{a_{dj}}{g(\boldsymbol{k}_j)} - \frac{1}{N}\sum_{j'=1}^{N} \frac{k(\boldsymbol{q}_i, \boldsymbol{k}_{j'})}{g(\boldsymbol{q}_i)} \sum_{j=1}^{N} \frac{a_{dj}}{g(\boldsymbol{k}_{j'})} \\
&= \sum_{j=1}^{N} \frac{k(\boldsymbol{q}_i, \boldsymbol{k}_j)}{g(\boldsymbol{q}_i)} \Big(\frac{a_{dj}}{g(\boldsymbol{k}_j)} - \frac{1}{N}\sum_{j'=1}^{N} \frac{a_{dj'}}{g(\boldsymbol{k}_j)}\Big) = \sum_{j=1}^{N} \frac{k(\boldsymbol{q}_i, \boldsymbol{k}_j)}{g(\boldsymbol{q}_i)} v_{dj},
\end{aligned} \tag{8}$$

where the kernel $k(\boldsymbol{x}, \boldsymbol{y}) := \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{y})$ and $v_{dj} := \frac{a_{dj}}{g(\boldsymbol{k}_j)} - \frac{1}{N}\sum_{j'=1}^{N} \frac{a_{dj'}}{g(\boldsymbol{k}_j)}$. We further let the self-attention's value vectors $\boldsymbol{v}_j = [v_{1j}, \cdots, v_{D_v j}] \in \mathbb{R}^{D_v \times 1}$, $j = 1, \dots, N$, and rewrite the projection $\boldsymbol{h}_i$ as $\boldsymbol{h}_i = \sum_{j=1}^{N} k(\boldsymbol{q}_i, \boldsymbol{k}_j)/g(\boldsymbol{q}_i)\boldsymbol{v}_j$. Selecting $g(\boldsymbol{x}) := \sum_{j=1}^{N} k(\boldsymbol{x}, \boldsymbol{k}_j)$ and $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(\boldsymbol{x}^\top \boldsymbol{y}/\sqrt{D})$, we obtain a formula of an attention:

$$\boldsymbol{h}_i = \sum_{j=1}^{N} \frac{k(\boldsymbol{q}_i, \boldsymbol{k}_j)}{\sum_{j'=1}^{N} k(\boldsymbol{q}_i, \boldsymbol{k}_{j'})}\boldsymbol{v}_j = \sum_{j=1}^{N} \mathrm{softmax}\Big(\boldsymbol{q}_i^\top \boldsymbol{k}_j/\sqrt{D}\Big)\boldsymbol{v}_j. \tag{9}$$

Recovering Self-Attention: Eqn. (9) matches the exact formula of a self-attention as in Eqn. (3). Thus, we can view outputs $\boldsymbol{h}_i$ of self-attention as projections of the query vectors $\boldsymbol{q}_i$, $i = 1, \dots, N$, onto D_v principal components in a feature space:

$$\boldsymbol{H} = [\boldsymbol{\varphi}(\boldsymbol{q}_1), \dots, \boldsymbol{\varphi}(\boldsymbol{q}_N)]^\top [\boldsymbol{u}_1, \dots, \boldsymbol{u}_{D_v}]. \tag{10}$$

Computing the Value Vectors: As derived above, the self-attention's value vectors $\boldsymbol{v}_j$ are given by: $\boldsymbol{v}_j = [v_{1j}, \cdots, v_{D_v j}] \in \mathbb{R}^{D_v \times 1}$, $j = 1, \dots, N$, where $v_{dj} := \frac{a_{dj}}{g(\boldsymbol{k}_j)} - \frac{1}{N}\sum_{j'=1}^{N} \frac{a_{dj'}}{g(\boldsymbol{k}_j)}$, $d = 1, \dots, D_v$. Since $g(\boldsymbol{k}_j)$ can be calculated as $g(\boldsymbol{k}_j) = \sum_{j'=1}^{N} k(\boldsymbol{k}_j, \boldsymbol{k}_{j'})$, in order to compute $\boldsymbol{v}_j$, we need to determine the coefficients $a_{1j}, \dots, a_{D_v j}$ for $j = 1, \dots, N$.

We define $\tilde{k}_{\boldsymbol{\varphi}}(\boldsymbol{x}, \boldsymbol{y}) := \tilde{\boldsymbol{\varphi}}(\boldsymbol{x})^\top \tilde{\boldsymbol{\varphi}}(\boldsymbol{y})$ and the Gram matrix $\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}} \in \mathbb{R}^{N \times N}$ with elements $\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}(i, j) = \tilde{k}_{\boldsymbol{\varphi}}(\boldsymbol{k}_i, \boldsymbol{k}_j)$. Substituting the linear expansion in Eqn. (7) into (6), we attain

$$\frac{1}{N}\sum_{j=1}^{N} \tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_j)\tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_j)^\top \sum_{j'=1}^{N} a_{dj'}\tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_{j'}) = \lambda_d \sum_{j=1}^{N} a_{dj}\tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_j).$$

We multiply both sides of the above by $\tilde{\varphi}(\boldsymbol{k}_\ell)^\top$ to obtain $\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}^2\boldsymbol{a}_d = \lambda_d N\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}\boldsymbol{a}_d$, with the column vector $\boldsymbol{a}_d = [a_{d1}, \cdots, a_{dN}]^\top \in \mathbb{R}^{N\times 1}$. We compute $\boldsymbol{a}_d$ by solving

$$\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}\boldsymbol{a}_d = \lambda_d N\boldsymbol{a}_d. \tag{11}$$

The calculation of $\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}$ is provided in Remark 1. We summarize our results in the following theorem.

Theorem 1 (Softmax Attention as Principal Component Projections) *Given a set M of key vectors, $M := \{\boldsymbol{k}_1, \ldots, \boldsymbol{k}_N\} \subset \mathbb{R}^D$, a kernel $k(\boldsymbol{x}, \boldsymbol{y}) := \exp(\boldsymbol{x}^\top\boldsymbol{y}/\sqrt{D})$, and a vector-scalar function $g(\boldsymbol{x}) := \sum_{j=1}^N k(\boldsymbol{x}, \boldsymbol{k}_j)$, self-attention performs kernel PCA and projects a query vector $\boldsymbol{q}_i \in \mathbb{R}^D$ onto principal component axes of M in an infinite dimensional feature space $\boldsymbol{\varphi}$ as follows*

$$\boldsymbol{h}_i = \sum_{j=1}^N \mathrm{softmax}\Big(\boldsymbol{q}_i^\top\boldsymbol{k}_j/\sqrt{D}\Big)\boldsymbol{v}_j.$$

The feature space $\boldsymbol{\varphi}$ is induced by the kernel $k_{\boldsymbol{\varphi}}(\boldsymbol{x}, \boldsymbol{y}) := \frac{k(\boldsymbol{x},\boldsymbol{y})}{g(\boldsymbol{x})g(\boldsymbol{y})}$, and the value vectors $\boldsymbol{v}_j = [v_{1j}, \ldots, v_{D_v j}] \in \mathbb{R}^{D_v\times 1}$, $j = 1, \ldots, N$, where $v_{dj} := \frac{a_{dj}}{g(\boldsymbol{k}_j)} - \frac{1}{N}\sum_{j'=1}^N \frac{a_{dj'}}{g(\boldsymbol{k}_j)}$, $d = 1, \ldots, D_v$. The column vectors $\boldsymbol{a}_d = [a_{d1}, \ldots, a_{dN}]^\top \in \mathbb{R}^{N\times 1}$ can be determined by solving the eigenvalue problem defined in Eqn. (11). This constraint on $\boldsymbol{v}_j$ can be relaxed by letting the self-attention learn $\boldsymbol{v}_j$ from data via a linear projection of the input $\boldsymbol{x}_j$, i.e., $\boldsymbol{v}_j = \boldsymbol{W}_V\boldsymbol{x}_j$ where $\boldsymbol{W}_V$ is a learnable matrix.

Remark 1 (Calculating the Gram Matrix $\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}$) In the eigenvalue problem defined in Eqn. (11), the centered Gram matrix $\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}$ can be computed from the uncentered Gram matrix $\boldsymbol{K}_{\boldsymbol{\varphi}}$ with elements $\boldsymbol{K}_{\boldsymbol{\varphi}}(i, j) = k_{\boldsymbol{\varphi}}(\boldsymbol{k}_i, \boldsymbol{k}_j) = \boldsymbol{\varphi}(\boldsymbol{k}_i)^\top\boldsymbol{\varphi}(\boldsymbol{k}_j)$. In particular, $\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}} = \boldsymbol{K}_{\boldsymbol{\varphi}} - \boldsymbol{1}_N\boldsymbol{K}_{\boldsymbol{\varphi}} - \boldsymbol{K}_{\boldsymbol{\varphi}}\boldsymbol{1}_N + \boldsymbol{1}_N\boldsymbol{K}_{\boldsymbol{\varphi}}\boldsymbol{1}_N$, where $\boldsymbol{1}_N$ denotes the $N\times N$ matrix in which every element takes the value $1/N$ [8] (See Appendix B). Here, the kernel $k_{\boldsymbol{\varphi}}(\boldsymbol{x}, \boldsymbol{y}) := \frac{k(\boldsymbol{x},\boldsymbol{y})}{g(\boldsymbol{x})g(\boldsymbol{y})}$.

Remark 2 (Determining D_v) The feature space $\boldsymbol{\varphi}$ is infinite dimensional, so we can find infinitely many principal components. However, the number of nonzero eigenvalues of $\boldsymbol{C}$ in Eqn. (5) cannot exceed N, the number of data points, since $\boldsymbol{C}$ has rank at most equal to N. Notice that only principal components corresponding to nonzero eigenvalues are used for projections in kernel PCA. Thus, D_v, the number of principal components used for projections as in Eqn. (10), must be less than or equal to N, i.e., $D_v \le N$.

Remark 3 (Parameterization of the Value Matrix $\boldsymbol{V}$) Different parameterizations of the value matrix $\boldsymbol{V}$ can result in different self-attention architectures. For instance, the projection $\boldsymbol{h}_i$ of a query vector $\boldsymbol{q}_i$ onto the principal components $\boldsymbol{u}_d$, $d = 1, \ldots, D_v$, in Eqn. (8) can be rewritten as

$$\boldsymbol{h}_i(d) = \sum_{j=1}^N \frac{k(\boldsymbol{q}_i, \boldsymbol{k}_j)}{g(\boldsymbol{q}_i)}\Big(\frac{a_{dj}}{g(\boldsymbol{k}_j)} - \frac{1}{N}\sum_{j'=1}^N \frac{g(\boldsymbol{k}_{j'})}{g(\boldsymbol{k}_j)}\frac{a_{dj'}}{g(\boldsymbol{k}_{j'})}\Big).$$

Letting $v_{dj} := \frac{a_{dj}}{g(\boldsymbol{k}_j)}$ and $s_{jj'} = \frac{g(\boldsymbol{k}_{j'})}{g(\boldsymbol{k}_j)}$, we obtain

$$\boldsymbol{h}_i(d) = \sum_{j=1}^N \frac{k(\boldsymbol{q}_i, \boldsymbol{k}_j)}{g(\boldsymbol{q}_i)}\Big(v_{dj} - \frac{1}{N}\sum_{j'=1}^N s_{jj'}v_{dj'}\Big).$$

Following the same derivation as above, we can write the projection $\boldsymbol{h}_i$ as an attention

$$\boldsymbol{h}_i = \sum_{j=1}^N \mathrm{softmax}\Big(\boldsymbol{q}_i^\top\boldsymbol{k}_j/\sqrt{D}\Big)\Big(\boldsymbol{v}_j - \frac{1}{N}\sum_{j'=1}^N s_{jj'}\boldsymbol{v}_{j'}\Big).$$

The matrix form of this new attention form is as follows:

$$\boldsymbol{H} = \mathrm{softmax}\Big(\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{D}\Big)(\boldsymbol{I} - \boldsymbol{S})\boldsymbol{V}, \tag{12}$$

where $\boldsymbol{I}$ is an identity matrix, and $\boldsymbol{S}$ is the matrix whose elements $\boldsymbol{S}(j, j') = \frac{1}{N}s_{jj'}$. We name the self-attention defined by Eqn. (12) the Scaled Attention. Even though the softmax attention in (2) and the Scaled Attention in (12) are mathematically equivalent according to our kernel PCA framework, the training procedure might cause the self-attention models that are derived from different parameterizations to have different performances.

https://doi.org/10.52202/079017-3215

2.2 Analysis on the Convergence of Self-Attention Layers to Kernel PCA

In this section, we discuss empirical justifications that after training, the value vectors $\boldsymbol{v}_j$ parameterized as a 1-layer linear network, i.e., $\boldsymbol{v}_j = \boldsymbol{W}_V \boldsymbol{x}_j$, $j = 1, \dots, N$, in self-attention converge to the values predicted by our kernel PCA framework in Theorem 1. In other words, we provide evidence that the self-attention layers in transformers try to learn their value vectors $\boldsymbol{v}_j$ to perform the kernel PCA.

2.2.1 Projection Error Minimization

PCA can be formulated based on projection error minimization as well. In particular, PCA minimizes the average projection cost defined as the mean squared distance between the original data points and their projections [62]. Given our kernel PCA framework in Theorem 1, this implies that self-attention minimizes the following projection error

$$J_{\text{proj}} = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{\varphi}(\boldsymbol{q}_i) - \sum_{d=1}^{D_v} h_{di} \boldsymbol{u}_d\|^2 = \frac{1}{N} \sum_{i=1}^{N} \left(\|\boldsymbol{\varphi}(\boldsymbol{q}_i)\|^2 - \|\boldsymbol{h}_i\|^2 \right). \tag{13}$$

In the derivation above, we leverage the orthonormality of $\{\boldsymbol{u}_d\}_{d=1}^{D_v}$ and $h_{di} = \boldsymbol{\varphi}(\boldsymbol{q}_i)^\top \boldsymbol{u}_d$. Here, notice that we can compute $\|\boldsymbol{\varphi}(\boldsymbol{q}_i)\|^2$ from $\boldsymbol{q}_i$ and $\{\boldsymbol{k}_j\}_{j=1}^{N}$ as $\|\boldsymbol{\varphi}(\boldsymbol{q}_i)\|^2 = \exp(\boldsymbol{q}_i^\top \boldsymbol{q}_i / \sqrt{D}) / (\sum_{j=1}^{N} \exp(\boldsymbol{q}_i^\top \boldsymbol{k}_j / \sqrt{D}))^2$. In Fig. 1, we empirically show that a transformer model minimizes the projection loss J_{proj} during training. Here, we train a vision transformer [22], ViT-tiny model in particular, on the ImageNet-1K classification task and compute the average of J_{proj} across attention heads and layers. This result suggests that during training, the transformer learns to perform PCA at each self-attention layer by implicitly minimizing the projection loss J_{proj}. Thus, the value vector $\boldsymbol{v}_j$, $j = 1, \dots, N$, in self-attention layers converge to the values specified in Theorem 1. We provide more details on the computation of J_{proj} in Appendix C.

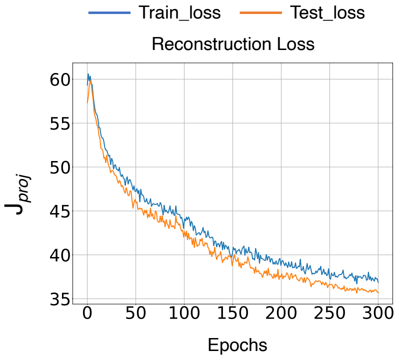

Figure 1: Projection loss vs. training epochs of ViT-tiny model. The reconstruction loss is averaged over the batch, heads, and layers. The downward trend suggests that the model is implicitly minimizing this projection loss.

2.2.2 Learning Eigenvectors of $\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}$in Eqn. (11)

In this section, we study empirical results that confirm Eqn. (11). In particular, we aim to verify that after training, the value matrix $\boldsymbol{V} := [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N]^\top$ captures the eigenvectors $\boldsymbol{a}_d$, $d = 1, \dots, D_v$, of the Gram matrix $\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}$.

We first compute $\boldsymbol{a}_d$ in terms of $\boldsymbol{V}$. Recall from Eqn. (11) that $\boldsymbol{a}_d = [a_{d1}, a_{d2}, \cdots, a_{dN}]^\top$. We denote the diagonal matrix $\boldsymbol{G} := \text{diag}(1/g(\boldsymbol{k}_1), \dots, 1/g(\boldsymbol{k}_N))$, the matrix $\boldsymbol{A} := [\boldsymbol{a}_1, \dots, \boldsymbol{a}_{D_v}]$, and rewrite the value vectors $\boldsymbol{v}_j$, $j = 1, \dots, N$, as follows:

$$\boldsymbol{v}_j = \left[\frac{\boldsymbol{a}_1[j]}{g(\boldsymbol{k}_j)} - \frac{1}{N} \sum_{j'=1}^{N} \frac{\boldsymbol{a}_1[j']}{g(\boldsymbol{k}_j)}, \dots, \frac{\boldsymbol{a}_{D_v}[j]}{g(\boldsymbol{k}_j)} - \frac{1}{N} \sum_{j'=1}^{N} \frac{\boldsymbol{a}_{D_v}[j']}{g(\boldsymbol{k}_j)} \right].$$

The value matrix $\boldsymbol{V}$ in self-attention is then given by

$$\boldsymbol{V} = \boldsymbol{G}\boldsymbol{A} - \boldsymbol{G}\boldsymbol{1}_N \boldsymbol{A} \Leftrightarrow \boldsymbol{A} = (\boldsymbol{I} - \boldsymbol{1}_N)^{-1} \boldsymbol{G}^{-1} \boldsymbol{V}$$

Thus, given the value matrix $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V^\top$ that the self-attention learns after training, the estimation $\hat{\boldsymbol{a}}_d$ of $\boldsymbol{a}_d$ can be computed as

$$\hat{\boldsymbol{a}}_d = (\boldsymbol{I} - \boldsymbol{1}_N)^{-1} \boldsymbol{G}^{-1} \boldsymbol{V}[:, d].$$

We empirically verify that $\frac{\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}} \hat{\boldsymbol{a}}_d}{N \hat{\boldsymbol{a}}_d} = \boldsymbol{\gamma} = [\gamma_1, \dots, \gamma_N]$ where $\gamma_1 = \cdots = \gamma_N = \text{const}$, which confirms that $\hat{\boldsymbol{a}}_d$ is an eigenvector of $\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}$. In particular, in Fig. 2(Left), we plot the average pairwise

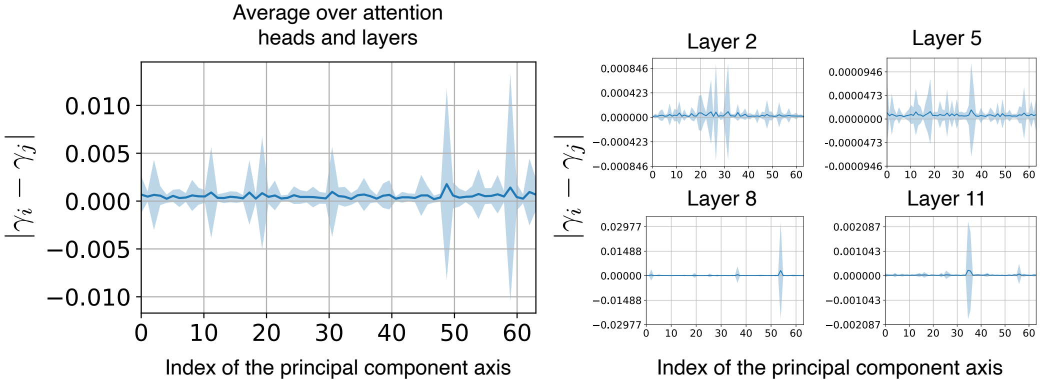

Figure 2: Mean and standard deviation of the absolute differences of elements in the constant vector $\mathbf{1}\lambda_d$, $d = 1, \dots, D_v$. The mean should be 0 with small standard deviations when v_{dj} are close to the values predicted in Theorem 1. For comparison, we observe that the max, min, mean, and median of the absolute values of all the eigenvalues, averaged over all attention heads and layers, are 648.46, 4.65, 40.07, and 17.73, respectively, which are much greater than the values of $|\gamma_i - \gamma_j|$.

absolute differences of γ_i and γ_j, $i \neq j$, $i, j = 1, \dots, N$, for each principal component axis of $\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}$. The results are averaged over all attention heads and all layers in the 12-layer ViT-tiny model trained on ImageNet-1K. As can be seen in our figure, the absolute difference between any pair of γ_i and γ_j is almost 0 with a very small standard deviation. Similar results are observed at each layer when averaging over all attention heads in that layer. In Fig. 2(Right), we show the results for Layers 2, 5, 8, and 11 in the model. For comparison, we observe that the max, min, mean, and median of the absolute values of these D_v eigenvalues, averaged over all attention heads and layers, are 648.46, 4.65, 40.07, and 17.73, respectively, which are much greater than the values of $|\gamma_i - \gamma_j|$. These results empirically justify that $\frac{\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}} \hat{\boldsymbol{a}}_d}{N \hat{\boldsymbol{a}}_d} = \text{const}$ and the value matrix $\boldsymbol{V}$ captures the eigenvectors of $\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}$ after the transformer model is trained, as suggested in Eqn. (11).

In order to prove that after training a transformer with the value vectors $\boldsymbol{v}_j$ parameterized as $\boldsymbol{W}_V \boldsymbol{x}_j$, $j = 1, \dots, N$, using stochastic gradient descent, v_{dj} converges to $\frac{a_{dj}}{g(\boldsymbol{k}_j)} - \frac{1}{N} \sum_{j'=1}^{N} \frac{a_{dj'}}{g(\boldsymbol{k}_j)}$ as stated in Theorem 1, it is sufficient to prove that after the training, the outputs $\boldsymbol{h}_i$ of self-attention become projections of the query vectors $\boldsymbol{q}_i$, $i = 1, \dots, N$, onto D_v principal component axes in the feature space $\boldsymbol{\varphi}$, i.e., the eigenvectors $\boldsymbol{u}_1, \dots, \boldsymbol{u}_{D_v}$ of the covariance matrix $\boldsymbol{C}$. To theoretically prove this result for a multi-layer multi-head softmax transformer trained to explicitly minimize a particular loss, e.g., cross-entropy or L2 loss, using stochastic gradient descent is indeed challenging due to the highly nonconvex nature of the optimization problem and the nonlinear structure of the model. Our experimental results in Section 2.2.1 and 2.2.2 empirically justify this result and serve as guidance for the theoretical proof, which we leave for future work.

3 Robust Softmax Attention

In this section, we employ our kernel PCA framework in Section 2 to derive a new class of robust attention, namely, *Attention with Robust Principal Components* (RPC-Attention). It is a well-known problem that both PCA and kernel PCA are sensitive to grossly corrupted data routinely encountered in modern applications [11, 49, 34, 19]. Since self-attention performs kernel PCA by projecting the query vectors $\boldsymbol{q}_i$, $i = 1, \dots, N$, onto principal components in a feature space as derived in Section 2.1, it is also vulnerable to data corruption and perturbation. Our RPC-Attention robustifies self-attention by solving a convex program known as Principal Component Pursuit (PCP) [11].

3.1 Principal Component Pursuit

Given corrupted measurement matrix $\boldsymbol{M} \in \mathbb{R}^{N \times D}$, both PCA and PCP aim to recover a low-rank matrix $\boldsymbol{L} \in \mathbb{R}^{N \times D}$ from $\boldsymbol{M}$. However, while PCA models the corruption by a small Gaussian noise term, PCP models the corruption by a matrix $\boldsymbol{S} \in \mathbb{R}^{N \times D}$ that can have arbitrarily large magnitude with sparse supports. In particular, PCP solves the following convex optimization problem:

$$\text{minimize}_{\boldsymbol{L},\boldsymbol{S}} \quad \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 \quad \text{subject to} \quad \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{M},$$

Algorithm 1 Principal Attention Pursuit (PAP)

```
initialize: $\boldsymbol{S}_0 = \boldsymbol{Y}_0 = \boldsymbol{0}$; $\mu, \lambda > 0$.
while not converged do
    compute $\boldsymbol{S}_{k+1} = \mathcal{S}_{\lambda/\mu}(\boldsymbol{K} - \boldsymbol{L}_k + \mu^{-1}\boldsymbol{Y}_k)$;
    compute $\boldsymbol{L}_{k+1} = \text{Softmax}(\boldsymbol{K} - \boldsymbol{S}_{k+1} - \mu^{-1}\boldsymbol{Y}_k, \boldsymbol{K} - \boldsymbol{S}_{k+1} - \mu^{-1}\boldsymbol{Y}_k)$;
    compute $\boldsymbol{Y}_{k+1} = \boldsymbol{Y}_k + \mu(\boldsymbol{K} - \boldsymbol{L}_{k+1} - \boldsymbol{S}_{k+1})$;
end while
output: $\boldsymbol{L}$.
```

where $\|\boldsymbol{L}\|_*$ is the nuclear norm of $\boldsymbol{L}$, i.e., the sum of the singular values of $\boldsymbol{L}$, and $\|\boldsymbol{S}\|_1 = \sum_{id} |S_{id}|$ is the ℓ_1-norm of $\boldsymbol{S}$. From [11], under minimal assumptions on the rank and sparsity of $\boldsymbol{L}$ and $\boldsymbol{S}$, the PCP solution exactly recovers the low-rank component $\boldsymbol{L}$ and the sparse component $\boldsymbol{S}$. *Since PCP can recover the principal components of a data matrix even when a positive fraction of the measurements are arbitrarily corrupted, it is more robust than PCA.*

3.2 Attention with Robust Principal Components

In self-attention, following our kernel PCA framework in Section 2, the dataset M is given as $M = \{\boldsymbol{k}_1, \dots, \boldsymbol{k}_N\} \subset \mathbb{R}^D$ and $\boldsymbol{k}_1, \dots, \boldsymbol{k}_N$ are key vectors. Thus, the key matrix $\boldsymbol{K} := [\boldsymbol{k}_1, \cdots, \boldsymbol{k}_N]^\top \in \mathbb{R}^{N \times D}$ in self-attention can be set as the measurement matrix $\boldsymbol{M}$ in PCP. Then, the PCP for self-attention can be formulated as

$$\text{minimize}_{\boldsymbol{L},\boldsymbol{S}} \quad \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 \quad \text{subject to} \quad \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{K}. \tag{14}$$

Following [11], we utilize the Alternating Direction Method of Multipliers (ADMM) algorithm introduced in [41, 91] to solve the convex program (14). The augmented Lagrangian of (14) is

$$\mathcal{L}(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{Y}) = \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 + \langle \boldsymbol{Y}, \boldsymbol{K} - \boldsymbol{L} - \boldsymbol{S}\rangle + \mu/2\|\boldsymbol{K} - \boldsymbol{L} - \boldsymbol{S}\|_F^2.$$

An ADMM solves the convex program (14) by iterating the following steps until convergence: [1] setting $\boldsymbol{S}_{k+1} = \arg\min_{\boldsymbol{S}} \mathcal{L}(\boldsymbol{L}_k, \boldsymbol{S}, \boldsymbol{Y}_k)$, [2] setting $\boldsymbol{L}_{k+1} = \arg\min_{\boldsymbol{L}} \mathcal{L}(\boldsymbol{L}, \boldsymbol{S}_{k+1}, \boldsymbol{Y}_k)$, and [3] updating the Lagrange multiplier matrix $\boldsymbol{Y}_{k+1} = \boldsymbol{Y}_k + \mu(\boldsymbol{K} - \boldsymbol{L}_{k+1} - \boldsymbol{S}_{k+1})$. We define $\mathcal{S}_\tau(x) := \text{sgn}(x)\max(|x| - \tau, 0)$ as an element-wise shrinkage operator and $\mathcal{D}_\tau(\boldsymbol{X}) = \boldsymbol{U}\mathcal{S}_\tau(\Sigma)\boldsymbol{V}^*$ as a singular value thresholding operator with the singular value decomposition of $\boldsymbol{X} = \boldsymbol{U}\Sigma\boldsymbol{V}^*$. As proven in [11], we can rewrite steps [1] and [2] as

$$\arg\min_{\boldsymbol{S}} \mathcal{L}(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{Y}) = \mathcal{S}_{\lambda/\mu}(\boldsymbol{K} - \boldsymbol{L} + \mu^{-1}\boldsymbol{Y}); \arg\min_{\boldsymbol{L}} \mathcal{L}(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{Y}) = \mathcal{D}_\mu(\boldsymbol{K} - \boldsymbol{S} - \mu^{-1}\boldsymbol{Y}).$$

$\mathcal{D}_\mu$ finds a low-rank approximation of $\boldsymbol{K} - \boldsymbol{S} - \mu^{-1}\boldsymbol{Y}$. Thus, we obtain an approximation of the above equation by replacing $\mathcal{D}_\mu$ by a low-rank approximation operator. Such an approximation takes a step towards the minimum of $\mathcal{L}(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{Y})$ when fixing $\boldsymbol{S}$ and $\boldsymbol{Y}$. It has been empirically observed and theoretically proven that the output matrix $\boldsymbol{H}$ of self-attention is low-rank, a phenomenon known as over-smoothing or rank collapse [71, 86, 21]. Therefore, we can replace $\mathcal{D}_\mu$ by a self-attention operator. The ADMM method applied to self-attention, which we name *Principal Attention Pursuit* (PAP), is given by Algorithm 1. We define our RPC-Attention as

Definition 1 (Attention with Robust Principal Components) *An RPC-Attention performs the PAP in Algorithm 1 for n iterations with λ as a hyperparameter. For the key matrix $\boldsymbol{K} \in \mathbb{R}^{N \times D}$, RPC-Attention sets $\mu = ND/4\|\boldsymbol{K}\|_1$ as suggested in [11], where $\|\boldsymbol{K}\|_1 = \sum_{id} |K_{id}|$. The output matrix $\boldsymbol{H}$ of RPC-Attention is set to be the low-rank output matrix $\boldsymbol{L}$ from PAP.*

4 Experimental Results

We aim to numerically show that: (i) RPC-Attention achieves competitive or even better accuracy than the baseline softmax attention on clean data, and (ii) the advantages of RPC-Attention are more prominent when there is a contamination of samples across different types of data and a variety of tasks. We also validate the performance of the Scaled Attention proposed in Remark 3.

Throughout our experiments, we compare the performance of our proposed models with the baseline softmax attention of the same configuration. All of our results are averaged over 5 runs with different seeds and run on 4 A100 GPU. Details on the models and training settings are provided in Appendix A and additional experimental results are provided in Appendix E. Primarily, we focus on a ViT-tiny model backbone [22], but included in the appendix are experiments on a larger model backbone, ViT-base, and a state of the art (SOTA) robust model, Fully Attentional Networks (FAN) [96].

Table 1: Top-1, Top-5 accuracy (%) , mean corruption error (mCE), and area under the precision-recall curve (AUPR) of RPC-SymViT and SymViT on clean ImageNet-1K data and popular standard robustness benchmarks for image classification. RPC-SymViT ($n$iter/layer1) applies n PAP iterations only at the first layer. RPC-SymViT ($n$iter/all-layer) applies n PAP iterations at all layers.

Model	IN-1K		IN-R	IN-A	IN-C		IN-O
	Top-1 ↑	Top-5 ↑	Top-1 ↑	Top-1 ↑	Top-1 ↑	mCE ↓	AUPR ↑
SymViT (baseline)	70.44	90.17	28.98	6.51	41.45	74.75	17.43
RPC-SymViT (4iter/layer1)	70.94	90.47	29.99	6.96	42.35	73.58	19.32
RPC-SymViT (5iter/layer1)	71.31	90.59	**30.28**	7.27	42.43	73.43	**20.35**
RPC-SymViT (6iter/layer1)	**71.49**	**90.68**	30.03	7.33	**42.76**	**73.03**	20.29
RPC-SymViT (2iter/all-layer)	70.59	90.15	29.23	**7.55**	41.64	74.52	19.18

Table 2: Top-1/5 accuracy (%) on attacked ImageNet-1K data. RPC-SymViT ($n$iter/layer1) applies n PAP iterations at the first layer. RPC-SymViT ($n$iter/all-layer) applies n PAP iterations at all layers.

Attack	Metric/Model	SymViT (baseline)	RPC-SymViT (4iter/layer1)	RPC-SymViT (5iter/layer1)	RPC-SymViT (6iter/layer1)	RPC-SymViT (2iter/all-layer)
PGD	Top-1 ↑	4.98	5.15	5.11	5.20	**6.12**
	Top-5 ↑	10.41	11.20	11.13	11.34	**13.24**
FGSM	Top-1 ↑	23.38	26.62	26.75	27.22	**29.20**
	Top-5 ↑	53.82	56.87	57.19	57.55	**59.63**
SPSA	Top-1 ↑	47.94	48.13	49.29	48.75	**51.01**
	Top-5 ↑	82.63	82.87	83.52	83.27	**83.66**
SLD	Top-1 ↑	67.91	68.48	68.60	**68.99**	67.60
	Top-5 ↑	89.68	90.16	90.18	**90.38**	89.79
CW	Top-1 ↑	48.44	50.00	50.08	**50.36**	48.77
	Top-5 ↑	72.68	74.24	74.14	**74.46**	72.91
Noise	Top-1 ↑	67.81	68.37	68.72	**68.81**	68.09
	Top-5 ↑	88.51	89.00	89.13	**89.27**	88.64
AutoAttack	Top-1 ↑	23.09	24.56	24.68	**24.74**	23.51
	Top-5 ↑	63.48	65.11	65.09	**65.13**	64.06

4.1 Vision Tasks: ImageNet-1K Object Classification

We implement PAP in Algorithm 1 in the symmetric softmax attention layers of a ViT-tiny model and compare it to the standard symmetric model as our baseline. We refer to our model as RPC-SymViT and the baseline model as SymViT. For RPC-SymViT, we study two settings. In the first setting, which we denote by RPC-SymViT ($n$iter/layer1), $n = 4, 5, 6$, to maintain the computational efficiency of the model, we apply n PAP iterations only at the first layer to recover a cleaner data matrix that is then sent to the subsequent layers in the model. In the second setting, which we denote by RPC-SymViT ($n$iter/all-layer), $n = 2$, we apply n PAP iterations at all layers. We note that an iterative scheme has the potential to have an increased computational load, hence, we provide a comparison on the number of flops per sample, run time per sample, memory and number of parameters between RPC-SymViT and the baseline in Appendix E.9, showing a comparable efficiency.

Robustness against Data Corruptions. To benchmark robustness to data corruptions, we use the standard datasets, ImageNet-C (IN-C) [31], ImageNet-A (IN-A), ImageNet-O (IN-O) [32], and ImageNet-R (IN-R) [30]. We provide details on each dataset and the metrics for evaluation in Appendix A.1. The direction of increasing or decreasing values of these metrics signifying greater robustness are indicated in the Table 1 with an arrow, along with the results on each dataset.

Across all evaluations, RPC-SymViT outperforms the SymViT baseline, thereby justifying the advantages of our RPC-Attention. Particularly, RPC-SymViT with 6 iterations in the 1st layer achieves an improvement of over 1% in terms of accuracy on the clean ImageNet-1K validation set and almost 3 AUPR on ImageNet-O compared to the SymViT. This result is consistent with the intuition that a higher number of iterations executed on a consistent data matrix leads to cleaner data. The full details of all corruption types are presented in Appendix E.3.

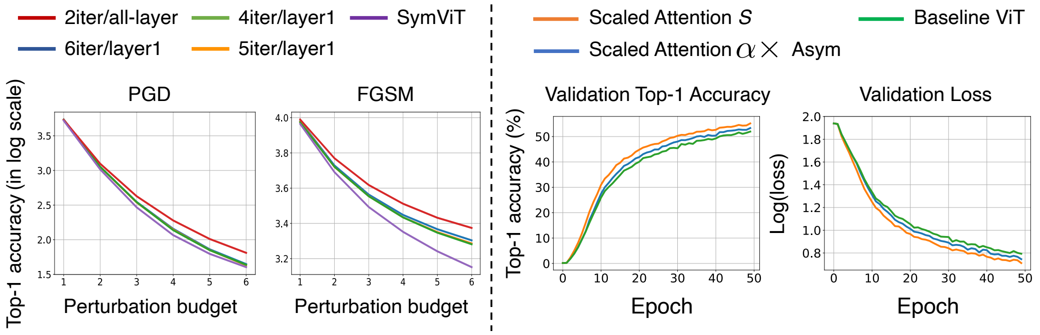

Figure 3: **Left:** Top-1 accuracy of RPC-SymViT vs. baseline SymViT evaluated on PGD/FGSM attacked ImageNet-1K validation set across increasing perturbation budgets. **Right:** Validation top-1 accuracy (%) and loss of Scaled Attention vs. the baseline asymmetric softmax attention in ViT for the first 50 training epochs.

Table 3: Validation/test perplexity (PPL) on clean WikiText-103 and word swap attacked dataset. RPC-Symlm ($n$iter/layern_1-n_2) applies n iterations of PAP only in layers n_1 to n_2 of the model.

Model	Wikitext-103		Attacked Wikitext-103	
	Valid-ppl ↓	Test-ppl ↓	Valid-ppl ↓	Test-ppl ↓
Symlm	38.37	36.36	42.90	44.32
RPC-Symlm (4iter/layer1-4)	**37.38**	**35.26**	**41.83**	**43.16**

Robustness under Adversarial Attacks. We report the top-1 and top-5 accuracy of the models on ImageNet-1K distorted by white box attacks, including PGD [45], FGSM [28], SLD [79], and CW [12] attacks. We further examine the models on the black-box attack SPSA [82], AutoAttack [18], and a noise-adding attack. In Table 2, we present the results of all attacks, where PGD and FGSM are reported for the maximum perturbation budget.

On all attacks, RPC-SymViT outperforms SymViT by a substantial margin, demonstrating the effectiveness of RPC-Attention. Notably, as the square attack in AutoAttack is also a score-based black box attack, in order to further verify our method, we include the results of RPC-SymViT (6iter/layer1) and the baseline SymViT on this attack in the Appendix E.4 in which our model also performs better. This result together with our model's considerable improvement over the baseline on the black-box SPSA attack justifies that RPC-Attention's robustness against adversarial attacks is not due to any form of gradient masking. In addition, we illustrate RPC-SymViT's robustness across increasing perturbations for PGD and FGSM in Fig. 3(left). Details on the evaluation of the models under all attacks are provided in Appendix A.2.

ADE20K Image Segmentation. We continue evaluating the benefits of our method by implementing RPC-Attention in a Segmenter model [72] and providing results on the ADE20K image segmentation task [95]. Table 6 in Appendix A.3 shows that Segmenter with the RPC-SymViT backbone outperforms the Segmenter with the baseline SymViT backbone on both clean and corrupted data.

4.2 Language Tasks: WikiText-103 Language Modeling

We assess our model on the large-scale WikiText-103 language modeling task [47]. Using a standard transformer language model [83], with a symmetric attention layer (Symlm), we replace it with an RPC-Attention layer (RPC-Symlm). As with RPC-SymViT, we only implement RPC-Attention in the first four layers of Symlm and run PAP for 4 iterations to save on computational overhead. The results of the validation and test perplexity (PPL) summarized in Table 3 validate the advantage of our method, surpassing the baseline by at least 1 PPL for all datasets.

4.3 Validating the Benefits of Scaled Attention

In this section, we provide an empirical comparison between the Scaled Attention in Remark 3 and the softmax attention. We train an asymmetric ViT-tiny model with two different versions of Scaled Attention. While the exact value of $\boldsymbol{S}$ can be mathematically formulated as in Remark 3, it might lead to numerical errors that are difficult to handle. Therefore, in the first version of Scaled Attention, we let $\boldsymbol{S}$ be a learned parameter matrix in each layer (Scaled Attention S in Fig. 3(right)). In the second version of Scaled Attention (Scaled Attention $\alpha \times$ Asym in Fig. 3(right)), given that $s_{jj'} = g(k_{j'})/g(k_j)$, we rewrite $\boldsymbol{S}$ as the product of a symmetric softmax attention matrix and the reciprocal of its transpose. The model learns this reciprocal by a learnable scalar α and we let $\boldsymbol{S} = \alpha \boldsymbol{A}_{Sym}$, where $\boldsymbol{A}_{Sym}$ is the symmetric softmax attention. More details are in Appendix E.1.

Fig. 3(right) shows the top-1 validation accuracy and loss over 50 epochs when training ViT models equipped with Scaled Attention and softmax attention on the ImageNet-1K object classification task. The full training curve can also be found in Appendix E.1. The results suggest that both versions of Scaled Attention outperform the softmax attention. This provides further evidence that self-attention learns to approximate a kernel PCA since the Scaled Attention with a more explicit structure of the value matrix $\boldsymbol{V}$ suggested in Theorem 1 obtains better performance than softmax attention.

5 Related Works

Theoretical Perspectives of Attention Mechanisms. The study of the attention mechanism in transformers through different theoretical frameworks has been expanding. [81] shows that attention can be analyzed as a kernel smoother over the inputs using an appropriate kernel score that is the similarities between them. [16, 87, 37, 55] reduce the quadratic complexity of transformers by linearizing the softmax kernel to improve the computational and memory efficiency. In addition, there are works interpreting transformers using the frameworks of ordinary/partial differential equations [50, 51, 44, 69, 27, 26] and from probabilistic viewpoints with Gaussian Mixture Models [52, 23, 75, 54, 92]. [15] provides a new perspective by emphasizing the asymmetry of the softmax kernel and recovers the self-attention mechanism from an asymmetric Kernel Singular Value Decomposition (KSVD) using the duality of the optimization problem. Another related work views transformers from the perspective of Support Vector Machines [57, 76]. We discuss [15] and [76], as well as the approaches that use matrix decomposition and iterative algorithms in deep models, in Appendix F. Separate from these works, our kernel PCA perspective derives softmax attention as a projection of the query vectors in a feature space. Using our framework, we are able to predict the exact explicit form of the value matrix in self-attention, demonstrating that this matrix captures the eigenvectors of the Gram matrix of the key vectors in a feature space. Our work is the first to show this insight.

Robustness of Transformers. There have been many works studying the robustness of Vision Transformers (ViT) [22] against different types of attacks [7, 61, 73, 96]. Recent work that serves to address this include [46], whereby new training strategies and architectural adjustments are proposed to improve the robustness of ViT. In addition, [58] suggests employing a Mahalanobis distance metric to calculate attention weights, expanding the feature space along directions of high contextual relevance, thereby enhancing the model's robustness. Also, [29] adapts traditional robust kernel density estimation techniques to create new classes of transformers that are resilient to adversarial attacks and data contamination. [14, 10] integrate a Gaussian process into attention for out-of-distribution detection, and [80] develops equivariant neural functional networks for transformers. Our RPC-Attention is orthogonal to these methods.

6 Concluding Remarks

In this paper, we derive self-attention from kernel principal component analysis (kernel PCA) as a projection of the query vectors onto the principal component axes of the key matrix in a feature space. Using our kernel PCA framework, we derive a new class of robust attention, namely the Attention with Robust Principal Components (RPC-Attention), that is resilient to data contamination. A limitation of RPC-Attention is its derivation from an iterative algorithm that leads to its unrolling architecture, increasing the computational cost of the model. In our paper, we mitigate this by only replacing softmax attention with RPC-Attention in the first layer of the model and demonstrate that doing so is sufficiently effective for robustness. In addition, we provide a comparison of the efficiency of RPC-Attention with softmax attention in Appendix E.9 and find that we are comparable across all metrics at test time while only slightly less efficient during training. It is also interesting to extend our kernel PCA framework to explain multi-layer transformers. We leave these exciting directions as future work.

Acknowledgments and Disclosure of Funding

This research / project is supported by the National Research Foundation Singapore under the AI Singapore Programme (AISG Award No: AISG2-TC-2023-012-SGIL). This research / project is supported by the Ministry of Education, Singapore, under the Academic Research Fund Tier 1 (FY2023) (A-8002040-00-00, A-8002039-00-00). This research / project is also supported by the NUS Presidential Young Professorship Award (A-0009807-01-00).

References

[1] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3159–3166, 2019.

[2] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pages 136–145. PMLR, 2017.

[3] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European conference on computer vision*, pages 484–501. Springer, 2020.

[4] Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*, 2019.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

[6] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in neural information processing systems*, 32, 2019.

[7] Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. Understanding robustness of transformers for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10231–10241, 2021.

[8] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[10] Long Minh Bui, Tho Tran Huu, Duy Dinh, Tan Minh Nguyen, and Trong Nghia Hoang. Revisiting kernel attention with correlated gaussian process representation. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024.

[11] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):1–37, 2011.

[12] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[13] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[14] Wenlong Chen and Yingzhen Li. Calibrating transformers via sparse gaussian processes. In *The Eleventh International Conference on Learning Representations*, 2023.

https://doi.org/10.52202/079017-3215

[15] Yingyi Chen, Qinghua Tao, Francesco Tonin, and Johan Suykens. Primal-attention: Self-attention through asymmetric kernel svd in primal representation. *Advances in Neural Information Processing Systems*, 36, 2024.

[16] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

[17] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[18] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020.

[19] Fernando De la Torre and Michael J Black. Robust principal component analysis for computer vision. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 362–369. IEEE, 2001.

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

[21] 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*, pages 2793–2803. PMLR, 2021.

[22] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[23] Prasad Gabbur, Manjot Bilkhu, and Javier Movellan. Probabilistic attention for interactive segmentation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 4448–4460. Curran Associates, Inc., 2021.

[24] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit Haim Bermano, Gal Chechik, and Daniel Cohen-or. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *The Eleventh International Conference on Learning Representations*, 2023.

[25] Zhengyang Geng, Meng-Hao Guo, Hongxu Chen, Xia Li, Ke Wei, and Zhouchen Lin. Is attention better than matrix decomposition? In *International Conference on Learning Representations*, 2021.

[26] Borjan Geshkovski, Cyril Letrouit, Yury Polyanskiy, and Philippe Rigollet. A mathematical perspective on transformers. *arXiv preprint arXiv:2312.10794*, 2023.

https://doi.org/10.52202/079017-3215

[27] Borjan Geshkovski, Cyril Letrouit, Yury Polyanskiy, and Philippe Rigollet. The emergence of clusters in self-attention dynamics. *Advances in Neural Information Processing Systems*, 36, 2024.

[28] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[29] Xing Han, Tongzheng Ren, Tan Nguyen, Khai Nguyen, Joydeep Ghosh, and Nhat Ho. Designing robust transformers using robust kernel density estimation. *Advances in Neural Information Processing Systems*, 36, 2024.

[30] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8340–8349, 2021.

[31] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

[32] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15262–15271, 2021.

[33] John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China, November 2019. Association for Computational Linguistics.

[34] Mia Hubert, Peter J Rousseeuw, and Karlien Vanden Branden. Robpca: a new approach to robust principal component analysis. *Technometrics*, 47(1):64–79, 2005.

[35] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[36] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[37] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[38] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

[39] Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc., 2022.

[40] Kuang-Huei Lee, Ofir Nachum, Mengjiao Sherry Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35:27921–27936, 2022.

[41] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

[42] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *International Conference on Learning Representations*, 2017.

[43] 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.

[44] Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. Understanding and improving transformer from a multi-particle dynamic system point of view. *arXiv preprint arXiv:1906.02762*, 2019.

[45] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[46] Xiaofeng Mao, Gege Qi, Yuefeng Chen, Xiaodan Li, Ranjie Duan, Shaokai Ye, Yuan He, and Hui Xue. Towards robust vision transformer. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 12042–12051, 2022.

[47] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

[48] John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, 2020.

[49] Minh Nguyen and Fernando Torre. Robust kernel principal component analysis. *Advances in Neural Information Processing Systems*, 21, 2008.

[50] Tam Nguyen, Tan Nguyen, and Richard Baraniuk. Mitigating over-smoothing in transformers via regularized nonlocal functionals. *Advances in Neural Information Processing Systems*, 36:80233–80256, 2023.

[51] Tam Nguyen, César A Uribe, Tan M Nguyen, and Richard G Baraniuk. Pidformer: Transformer meets control theory. In *International Conference on Machine Learning*. PMLR, 2024.

[52] Tam Minh Nguyen, Tan Minh Nguyen, Dung DD Le, Duy Khuong Nguyen, Viet-Anh Tran, Richard Baraniuk, Nhat Ho, and Stanley Osher. Improving transformers with probabilistic attention keys. In *International Conference on Machine Learning*, pages 16595–16621. PMLR, 2022.

[53] Tan Nguyen, Tam Nguyen, Hai Do, Khai Nguyen, Vishwanath Saragadam, Minh Pham, Khuong Duy Nguyen, Nhat Ho, and Stanley Osher. Improving transformer with an admixture of attention heads. *Advances in neural information processing systems*, 35:27937–27952, 2022.

[54] Tan Nguyen, Minh Pham, Tam Nguyen, Khai Nguyen, Stanley Osher, and Nhat Ho. Fourierformer: Transformer meets generalized fourier integral theorem. *Advances in Neural Information Processing Systems*, 35:29319–29335, 2022.

[55] Tan Nguyen, Vai Suliafu, Stanley Osher, Long Chen, and Bao Wang. Fmmformer: Efficient and flexible transformer via decomposed near-field and far-field attention. *Advances in neural information processing systems*, 34:29449–29463, 2021.

[56] Tan M Nguyen, Tam Nguyen, Long Bui, Hai Do, Duy Khuong Nguyen, Dung D Le, Hung Tran-The, Nhat Ho, Stan J Osher, and Richard G Baraniuk. A probabilistic framework for pruning transformers via a finite admixture of keys. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[57] Tan Minh Nguyen, Tam Minh Nguyen, Nhat Ho, Andrea L. Bertozzi, Richard Baraniuk, and Stanley Osher. A primal-dual framework for transformers and neural networks. In *The Eleventh International Conference on Learning Representations*, 2023.

[58] Stefan K Nielsen, Laziz U Abdullaev, Rachel Teo, and Tan M Nguyen. Elliptical attention. *Advances in Neural Information Processing Systems*, 2024.

[59] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.

[60] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas, November 2016. Association for Computational Linguistics.

[61] Sayak Paul and Pin-Yu Chen. Vision transformers are robust learners. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 36, pages 2071–2081, 2022.

[62] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[63] 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*, pages 8748–8763. PMLR, 2021.

[64] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI report*, 2018.

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

[66] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[67] Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8844–8856. PMLR, 18–24 Jul 2021.

[68] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15), 2021.

[69] Michael E Sander, Pierre Ablin, Mathieu Blondel, and Gabriel Peyré. Sinkformers: Transformers with doubly stochastic attention. In *International Conference on Artificial Intelligence and Statistics*, pages 3515–3530. PMLR, 2022.

[70] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pages 9355–9366. PMLR, 2021.

[71] Han Shi, Jiahui Gao, Hang Xu, Xiaodan Liang, Zhenguo Li, Lingpeng Kong, Stephen M. S. Lee, and James Kwok. Revisiting over-smoothing in BERT from the perspective of graph. In *International Conference on Learning Representations*, 2022.

[72] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272, 2021.

[73] Akshayvarun Subramanya, Aniruddha Saha, Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Backdoor attacks on vision transformers. *arXiv preprint arXiv:2206.08477*, 2022.

https://doi.org/10.52202/079017-3215

[74] J.A.K. Suykens, T. Van Gestel, J. Vandewalle, and B. De Moor. A support vector machine formulation to pca analysis and its kernel version. *IEEE Transactions on Neural Networks*, 14(2):447–450, 2003.

[75] Binh Tang and David S. Matteson. Probabilistic transformer for time series analysis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[76] Davoud Ataee Tarzanagh, Yingcong Li, Christos Thrampoulidis, and Samet Oymak. Transformers as support vector machines. *arXiv preprint arXiv:2308.16898*, 2023.

[77] Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy, July 2019. Association for Computational Linguistics.

[78] 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.

[79] Florian Tramer and Dan Boneh. Adversarial training and robustness for multiple perturbations. *Advances in neural information processing systems*, 32, 2019.

[80] Viet-Hoang Tran, Thieu N Vo, An Nguyen The, Tho Tran Huu, Minh-Khoi Nguyen-Nhat, Thanh Tran, Duy-Tung Pham, and Tan Minh Nguyen. Equivariant neural functional networks for transformers. *arXiv preprint arXiv:2410.04209*, 2024.

[81] Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4344–4353, Hong Kong, China, November 2019. Association for Computational Linguistics.

[82] Jonathan Uesato, Brendan O'donoghue, Pushmeet Kohli, and Aaron Oord. Adversarial risk and the dangers of evaluating against weak attacks. In *International Conference on Machine Learning*, pages 5025–5034. PMLR, 2018.

[83] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[84] Jesse Vig and Yonatan Belinkov. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy, August 2019. Association for Computational Linguistics.

[85] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics.

[86] Peihao Wang, Wenqing Zheng, Tianlong Chen, and Zhangyang Wang. Anti-oversmoothing in deep vision transformers via the fourier domain analysis: From theory to practice. In *International Conference on Learning Representations*, 2022.

[87] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[88] Zifeng Wang and Jimeng Sun. Transtab: Learning transferable tabular transformers across tables. *Advances in Neural Information Processing Systems*, 35:2902–2915, 2022.

[89] Yongyi Yang, Tang Liu, Yangkun Wang, Jinjing Zhou, Quan Gan, Zhewei Wei, Zheng Zhang, Zengfeng Huang, and David Wipf. Graph neural networks inspired by classical iterative algorithms. In *International Conference on Machine Learning*, pages 11773–11783. PMLR, 2021.

[90] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[91] Xiaoming Yuan and Junfeng Yang. Sparse and low-rank matrix decomposition via alternating direction method. *Pacific Journal of Optimization*, 9:167, 2013.

[92] Shaolei Zhang and Yang Feng. Modeling concentrated cross-attention for neural machine translation with Gaussian mixture model. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1401–1411, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[93] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[94] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 27042–27059. PMLR, 17–23 Jul 2022.

[95] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5122–5130, 2017.

[96] Daquan Zhou, Zhiding Yu, Enze Xie, Chaowei Xiao, Animashree Anandkumar, Jiashi Feng, and Jose M Alvarez. Understanding the robustness in vision transformers. In *International Conference on Machine Learning*, pages 27378–27394. PMLR, 2022.

[97] Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. A robustly optimized BERT pre-training approach with post-training. In Sheng Li, Maosong Sun, Yang Liu, Hua Wu, Kang Liu, Wanxiang Che, Shizhu He, and Gaoqi Rao, editors, *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, Huhhot, China, August 2021. Chinese Information Processing Society of China.

Supplement to “Unveiling the Hidden Structure of Self-Attention via Kernel Principal Component Analysis”

Table of Contents

A Experiment Details

Implementation Details of RPC-SymViT: Our RPC-SymViT models have 5.2M parameters, the same as the SymViT baseline. We use a standard tiny configuration with 12 transformer layers, 3 attention heads per layer, and a model dimension of 192 and simply replace softmax attention with RPC-Attention. We follow the training settings as in [78] and their implementation is available at https://github.com/facebookresearch/deit. In a Segmenter model, we use the same RPC-SymViT setting to replace the baseline SymViT backbone. We follow the training details in [72] and their code is publicly available as well, https://github.com/rstrudel/segmenter.

There are 3 hyperparameters: 1) μ: this parameter controls the singular value thresholding operator in the PAP algorithm. We set μ to the recommended value given in Definition 1; 2) λ: this is a regularization parameter that controls the sparsity of the corruption matrix $\boldsymbol{S}$. We finetune λ for training and observe that RPC-SymViT with $\lambda = 3$ yields the best performance for models with 2 iterations per layer and $\lambda = 4$ yields the best performance for models with iterations only in the first layer; 3) n: the number of iterations of the PAP algorithm in a RPC-Attention layer.

https://doi.org/10.52202/079017-3215

Implementation Details of RPC-Symlm: For our language model, we use a standard transformer language model [83], with a symmetric attention layer. The model has a dimension of 128 for the keys, queries and values, while the training and evaluation context length is set at 256. There are 16 layers altogether and 8 heads per layer. Similarly, we replace the softmax attention with RPC-Attention only in the first four layers to save on computational overhead. There are the same 3 hyperparameters as in RPC-SymViT and we use the same value for μ, $\lambda = 4$ and $n = 4$. We also follow the standard training settings as in [70, 47] and the code base developed by [70], available here https://github.com/IDSIA/lmtool-fwp.

A.1 Robustness against Data Corruptions

Datasets: We use the ImageNet-1K dataset that contains 1.28M training images and 50K validation images. There are 1000 classes of images and the model learns an image classification task. For robustness to common corruptions, we use ImageNet-C (IN-C) [31] which consists of 15 different types of corruptions applied to the ImageNet-1K validation set with 5 levels of severity. To test robustness to both input data distribution shifts as well as label distribution shifts, we use ImageNet-A (IN-A) and ImageNet-O (IN-O) [32] respectively. Both of these datasets contain a 200 class subset of ImageNet-1K classes with adversarially filtered images. Finally, we test our model on ImageNet-R (IN-R) [30] which contains various artistic renditions of images. This evaluates the model's generalization ability to abstract visual renditions.

Metrics: On ImageNet-1K, ImageNet-A and ImageNet-R, we report the top-1 accuracies for all experiments. We include top-5 accuracies on ImageNet-1K. On ImageNet-C, the standard metric for evaluation is the mean Corruption Error (mCE). To calculate this, we average the top-1 error rate for each corruption type across the 5 levels of severity and divide them by AlexNet's average errors, then take the final average across all corruption types. We report the area under the precision-recall curve (AUPR) for ImageNet-O which requires anomaly scores. The score is obtained by taking the negative of the highest softmax probability output by the model. The direction of increasing or decreasing values of these metrics signifying greater robustness will be indicated in the table with an arrow.

A.2 Robustness under Adversarial Attacks

Attacks: We evaluate the robustness of our method against adversarial attacks using CleverHans [59] and AutoAttack [18]. All attacks are executed on ImageNet-1K's validation set, and each model is evaluated on the whole set. In particular, we use untargeted, white box attacks such as PGD, FGSM, SLD and CW. In addition, we provide results on gradient-free black box attack, SPSA, diverse AutoAttack, in the standard setting and a simple noise-adding attack. AutoAttack consists of untargeted APGD-CE, targeted APGD-DLR, targeted Fast Adaptive Boundary (FAB) and Square Attack. For further justification of the benefit of our method, we evaluate a variant of our model, RPC-SymViT (6iter/layer1) solely on the black-box Square Attack and show that we are robust against black box attacks as well. This is provided in Appendix E.4. We use a perturbation budget of $\epsilon = 1/255$ with the l_∞ norm to manipulate the images and evaluate each model with an incremental increase in perturbation for PGD and FGSM. In SPSA, we run 40 steps per attack with a perturbation budget of $\epsilon = 0.1$. PGD attack uses a step size of $\alpha = 0.15$ for 20 steps, where at each step the image is adjusted slightly to maximize the model's loss. Similarly, FGSM does this for a single step. For each attack, we report the top-1 and top-5 accuracy of the model on the distorted dataset with the maximum perturbation budget in Table 2 and across all different perturbations in Figure 3.

A.3 ADE20K Image Segmentation

Dataset: The ADE20K dataset includes complex scenes featuring highly detailed labels, making it one of the most challenging segmentation tasks. The training set contains 20,210 images spread across 150 distinct semantic categories. The validation set includes 2,000 images, while the test set comprises 3,352 images. The metrics report for this task are the Mean Accuracy (%) and Mean Intersection-Over-Union (IOU).

https://doi.org/10.52202/079017-3215

A.4 WikiText-103 Language Modeling

Dataset: The WikiText-103 dataset [47] is derived from Wikipedia articles and is designed to capture long-range contextual dependencies. The training set contains about 28,000 articles, with a total of 103 million words. Each article is divided into text blocks with approximately 3,600 words. The validation and test sets have 218,000 and 246,000 words, respectively, with both sets comprising 60 articles and totaling about 268,000 words. For evaluation, we use a batch size of 1 and apply a sliding window of length L to process the text sequences. When calculating perplexity (PPL), we focus only on the final position, except in the first segment, where we evaluate all positions. We corrupt the both validation and test datasets to demonstrate the robustness of RPC-Attention using TextAttack's word swap attack [48] to create the attacked WikiText-103 dataset. This adversarial attack randomly replaces words in the dataset with a generic "AAA" for evaluation making it difficult for the model to predict the next word in the sequence correctly.

B Calculating the Gram Matrix $\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}$

In this section, we will show that the centered Gram matrix $\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}$ can be computed from the uncentered Gram matrix $\boldsymbol{K}_{\boldsymbol{\varphi}}$ with elements $\boldsymbol{K}_{\boldsymbol{\varphi}}(i,j) = k_{\boldsymbol{\varphi}}(\boldsymbol{k}_i, \boldsymbol{k}_j) = \boldsymbol{\varphi}(\boldsymbol{k}_i)^\top \boldsymbol{\varphi}(\boldsymbol{k}_j)$. In particular, $\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}} = \boldsymbol{K}_{\boldsymbol{\varphi}} - \mathbf{1}_N \boldsymbol{K}_{\boldsymbol{\varphi}} - \boldsymbol{K}_{\boldsymbol{\varphi}} \mathbf{1}_N + \mathbf{1}_N \boldsymbol{K}_{\boldsymbol{\varphi}} \mathbf{1}_N$, where $\mathbf{1}_N$ denotes the $N \times N$ matrix in which every element takes the value $1/N$. Our centered feature vector has the form

$$\tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_j) = \boldsymbol{\varphi}(\boldsymbol{k}_j) - \frac{1}{N} \sum_{j'=1}^{N} \boldsymbol{\varphi}(\boldsymbol{k}_{j'}).$$

Then the elements of the centered Gram matrix are given as follows

$$\begin{aligned}
\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}(i,j) &= \tilde{k}_{\boldsymbol{\varphi}}(\boldsymbol{k}_i, \boldsymbol{k}_j) \\
&= \tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_i)^\top \tilde{\boldsymbol{\varphi}}(\boldsymbol{k}_j) \\
&= \boldsymbol{\varphi}(\boldsymbol{k}_i)^\top \boldsymbol{\varphi}(\boldsymbol{k}_j) - \frac{1}{N} \sum_{j'=1}^{N} \boldsymbol{\varphi}(\boldsymbol{k}_i)^\top \boldsymbol{\varphi}(\boldsymbol{k}_{j'}) - \frac{1}{N} \sum_{j'=1}^{N} \boldsymbol{\varphi}(\boldsymbol{k}_{j'})^\top \boldsymbol{\varphi}(\boldsymbol{k}_j) + \frac{1}{N^2} \sum_{j'=1}^{N} \sum_{l=1}^{N} \boldsymbol{\varphi}(\boldsymbol{k}_{j'})^\top \boldsymbol{\varphi}(\boldsymbol{k}_l) \\
&= k_{\boldsymbol{\varphi}}(\boldsymbol{k}_i, \boldsymbol{k}_j) - \frac{1}{N} \sum_{j'=1}^{N} k_{\boldsymbol{\varphi}}(\boldsymbol{k}_i, \boldsymbol{k}_{j'}) - \frac{1}{N} \sum_{j'=1}^{N} k_{\boldsymbol{\varphi}}(\boldsymbol{k}_{j'}, \boldsymbol{k}_j) + \frac{1}{N^2} \sum_{j'=1}^{N} \sum_{l=1}^{N} k_{\boldsymbol{\varphi}}(\boldsymbol{k}_{j'}, \boldsymbol{k}_l)
\end{aligned}$$

which expressed in matrix form, gives the result.

C Plotting J_{proj} in Section 2.2.1

In Section 2.2.1, we plotted the reconstruction loss at each epoch of training. Here, we provide the details of the calculation of this loss. From Eqn. (13),

$$\begin{aligned}
L &= \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{\varphi}(\boldsymbol{q}_i) - \sum_{d=1}^{D_v} h_{id} u_d\|^2 \\
&= \frac{1}{N} \sum_{i=1}^{N} (\|\boldsymbol{\varphi}(\boldsymbol{q}_i)\|^2 - \|\boldsymbol{h}_i\|^2).
\end{aligned}$$

In the above, $\boldsymbol{h}_i$ is simply our attention output and $\|\boldsymbol{\varphi}(\boldsymbol{q}_i)\|^2 = \boldsymbol{\varphi}(\boldsymbol{q}_i)^\top \boldsymbol{\varphi}(\boldsymbol{q}_i) = \frac{e^{\boldsymbol{q}_i^\top \boldsymbol{q}_i / \sqrt{D}}}{(\sum_{j=1}^{N} e^{\boldsymbol{q}_i^\top \boldsymbol{k}_j} / \sqrt{D})^2}$ for $i = 1, \ldots, N$, can be calculated using the kernel trick as follows. Let $\boldsymbol{A}$ be the softmax attention

https://doi.org/10.52202/079017-3215

matrix, $\boldsymbol{a}_1$ its first column and $\boldsymbol{a}_1^2$ denote the element-wise product.

$$
\begin{aligned}
log(\boldsymbol{a}_1^2) &= log(\begin{bmatrix} \frac{e^{2\boldsymbol{q}_1^\top \boldsymbol{k}_1/\sqrt{D}}}{(\sum_{j=1}^N e^{\boldsymbol{q}_1^\top \boldsymbol{k}_j/\sqrt{D}})^2} \\ \vdots \\ \frac{e^{2\boldsymbol{q}_N^\top \boldsymbol{k}_1/\sqrt{D}}}{(\sum_{j=1}^N e^{\boldsymbol{q}_N^\top \boldsymbol{k}_j/\sqrt{D}})^2} \end{bmatrix}) \\
&= log(\begin{bmatrix} e^{2\boldsymbol{q}_1^\top \boldsymbol{k}_1/\sqrt{D}} \\ \vdots \\ e^{2\boldsymbol{q}_N^\top \boldsymbol{k}_1/\sqrt{D}} \end{bmatrix}) - log(\begin{bmatrix} (\sum_{j=1}^N e^{\boldsymbol{q}_1^\top \boldsymbol{k}_j/\sqrt{D}})^2 \\ \vdots \\ (\sum_{j=1}^N e^{\boldsymbol{q}_N^\top \boldsymbol{k}_j/\sqrt{D}})^2 \end{bmatrix})
\end{aligned}
$$

$$
\begin{aligned}
\implies log(\begin{bmatrix} (\sum_{j=1}^N e^{\boldsymbol{q}_1^\top \boldsymbol{k}_j/\sqrt{D}})^2 \\ \vdots \\ (\sum_{j=1}^N e^{\boldsymbol{q}_N^\top \boldsymbol{k}_j/\sqrt{D}})^2 \end{bmatrix}) &= log(\begin{bmatrix} e^{2\boldsymbol{q}_1^\top \boldsymbol{k}_1/\sqrt{D}} \\ \vdots \\ e^{2\boldsymbol{q}_N^\top \boldsymbol{k}_1/\sqrt{D}} \end{bmatrix}) - log(\boldsymbol{a}_1^2) \\
&= \begin{bmatrix} 2\boldsymbol{q}_1^\top \boldsymbol{k}_1/\sqrt{D} \\ \vdots \\ 2\boldsymbol{q}_N^\top \boldsymbol{k}_1/\sqrt{D} \end{bmatrix} - log(\boldsymbol{a}_1^2)
\end{aligned}
$$

$$
\implies \begin{bmatrix} (\sum_{j=1}^N e^{\boldsymbol{q}_1^\top \boldsymbol{k}_j/\sqrt{D}})^2 \\ \vdots \\ (\sum_{j=1}^N e^{\boldsymbol{q}_N^\top \boldsymbol{k}_j/\sqrt{D}})^2 \end{bmatrix} = e^{\begin{bmatrix} 2\boldsymbol{q}_1^\top \boldsymbol{k}_1/\sqrt{D} \\ \vdots \\ 2\boldsymbol{q}_N^\top \boldsymbol{k}_1/\sqrt{D} \end{bmatrix} - log(\boldsymbol{a}_1^2)}
$$

Then,

$$
\begin{bmatrix} \|\boldsymbol{\varphi}(\boldsymbol{q}_1)\|^2 \\ \vdots \\ \|\boldsymbol{\varphi}(\boldsymbol{q}_N)\|^2 \end{bmatrix} = \begin{bmatrix} e^{\boldsymbol{q}_1^\top \boldsymbol{q}_1/\sqrt{D}} \\ \vdots \\ e^{\boldsymbol{q}_N^\top \boldsymbol{q}_N/\sqrt{D}} \end{bmatrix} / e^{\begin{bmatrix} 2\boldsymbol{q}_1^\top \boldsymbol{k}_1/\sqrt{D} \\ \vdots \\ 2\boldsymbol{q}_N^\top \boldsymbol{k}_1/\sqrt{D} \end{bmatrix} - log(\boldsymbol{a}_1^2)}
$$

D Principal Component Pursuit

Let $\boldsymbol{M}, \boldsymbol{L}, \boldsymbol{S} \in \mathbb{R}^{N\times D}$ be the matrix of our corrupted measurements, the low-rank matrix we seek to recover and a sparse corruption matrix respectively. The optimization problem we would like to solve is

$$
\begin{array}{ll} \text{minimize}_{\boldsymbol{L},\boldsymbol{S}} & \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 \\ \text{subject to} & \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{M} \end{array} \tag{15}
$$

Under minimal assumptions that the low-rank component $\boldsymbol{L}$ is not sparse (i.e., $\boldsymbol{L}$ satisfies the incoherence condition defined in [11]), and the sparse component $\boldsymbol{S}$ is not low-rank (i.e., the sparsity pattern of S is selected uniformly at random), we will follow the author's choice of algorithm to solve the PCP which is to use the Alternating Direction Method of Multipliers (ADMM).

The ADMM algorithm uses the augmented Lagrangian,

$$
l(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{Y}) = \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 + \langle \boldsymbol{Y}, \boldsymbol{M} - \boldsymbol{L} - \boldsymbol{S}\rangle + \mu/2\|\boldsymbol{M} - \boldsymbol{L} - \boldsymbol{S}\|_F^2
$$

and solves a sequence of optimization problems. We iterate through setting $\boldsymbol{S}_{k+1} = \arg\min_{\boldsymbol{S}} l(\boldsymbol{L}_k, \boldsymbol{S}, \boldsymbol{Y}_k)$ and $\boldsymbol{L}_{k+1} = \arg\min_{\boldsymbol{L}} l(\boldsymbol{L}, \boldsymbol{S}_{k+1}, \boldsymbol{Y}_k)$ before updating the Langrange multiplier matrix $\boldsymbol{Y}_{k+1} = \boldsymbol{Y}_k + \mu(\boldsymbol{M} - \boldsymbol{L}_{k+1} - \boldsymbol{S}_{k+1})$. The advantage of the algorithm is that it turns a complicated optimisation problem into sub problems that have straightforward and efficient solutions. Without much difficulty we can show that:

$$
\begin{aligned}
\arg\min_{\boldsymbol{S}} l(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{Y}) =& \mathcal{S}_{\lambda/\mu}(\boldsymbol{M} - \boldsymbol{L} + \mu^{-1}\boldsymbol{Y}) \\
\arg\min_{\boldsymbol{L}} l(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{Y}) =& \mathcal{D}_{\mu}(\boldsymbol{M} - \boldsymbol{S} - \mu^{-1}\boldsymbol{Y})
\end{aligned}
$$

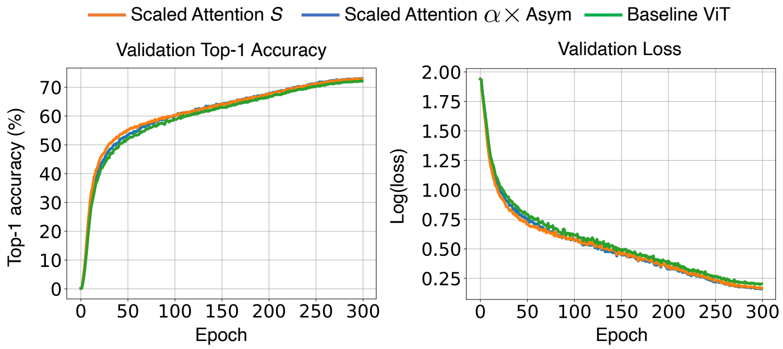

Figure 4: Plot of the validation top-1 accuracy (%) and loss on a log scale of the baseline asymmetric attention ViT and two variants with the parameterization of Remark. 3. The curves are plotted for the full training time and show $\boldsymbol{S}$ trained as a matrix parameter as well as a scalar parameter scaling a symmetric attention matrix.

where $\mathcal{S}_\tau(x) = \text{sgn}(x)\max(|x| - \tau, 0)$ is the shrinkage operator, extended to matrices by applying it element-wise and $\mathcal{D}_\tau(\boldsymbol{X}) = \boldsymbol{U}\mathcal{S}_\tau(\Sigma)\boldsymbol{V}^*$ is the singular value thresholding operator with the singular value decomposition of $\boldsymbol{X} = \boldsymbol{U}\Sigma\boldsymbol{V}^*$.

The algorithm is summarised as below

Algorithm 2 ADMM for Principal Components Pursuit

```
initialize: $\boldsymbol{S}_0 = \boldsymbol{Y}_0 = \mathbf{0}$; $\mu, \lambda > 0$.
while not converged do
    compute $\boldsymbol{S}_{k+1} = \mathcal{S}_{\lambda/\mu}(\boldsymbol{M} - \boldsymbol{L}_k + \mu^{-1}\boldsymbol{Y}_k)$;
    compute $\boldsymbol{L}_{k+1} = \mathcal{D}_{\mu}(\boldsymbol{M} - \boldsymbol{S}_{k+1} - \mu^{-1}\boldsymbol{Y}_k)$;
    compute $\boldsymbol{Y}_{k+1} = \boldsymbol{Y}_k + \mu(\boldsymbol{M} - \boldsymbol{L}_{k+1} - \boldsymbol{S}_{k+1})$;
end while
output: $\boldsymbol{L}$, $\boldsymbol{S}$.
```

E Additional Experimental Results

E.1 Reparameterization of Self-Attention

Let $\boldsymbol{A}_{sym} = [a_{ij}]$ be the softmax attention matrix. Then, from Remark 3, $\boldsymbol{S}$ has the following form and we multiply the numerator and denominator by $1/K(\boldsymbol{k}_j, \boldsymbol{k}_{j'})$ to obtain a much more convenient expression.

$$\begin{aligned}
s_{j'j} =& \frac{g(\boldsymbol{k}_j)}{g(\boldsymbol{k}_{j'})} \\
=& \frac{g(\boldsymbol{k}_j)}{g(\boldsymbol{k}_{j'})} \times \frac{1/K(\boldsymbol{k}_j, \boldsymbol{k}_{j'})}{1/K(\boldsymbol{k}_j, k_{\boldsymbol{j'}})} \\
=& \frac{g(\boldsymbol{k}_j)}{K(\boldsymbol{k}_j, \boldsymbol{k}_{j'})} \div \frac{g(\boldsymbol{k}_{j'})}{K(\boldsymbol{k}_j, \boldsymbol{k}_{j'})} \\
=& \frac{g(\boldsymbol{k}_j)}{K(\boldsymbol{k}_j, \boldsymbol{k}_{j'})} \times \frac{K(\boldsymbol{k}_j, \boldsymbol{k}_{j'})}{g(\boldsymbol{k}_{j'})} \\
=& \frac{a_{j'j}}{a_{jj'}} \\
\implies \boldsymbol{S} =& \frac{1}{N}\boldsymbol{A}_{sym} \odot 1/\boldsymbol{A}_{sym}^\top
\end{aligned}$$

In Fig. 4, we plot the full training curve of the two versions of Scaled Attention, Scaled Attention S and Scaled Attention $\alpha \times$ Asym. We observe that the parameterized models with Scaled Attention do converge more quickly and even obtain a slightly higher validation top-1 accuracy of 73.02% for the scalar variant as compared to the standard asymmetric ViT at 72.18%. The final validation loss is also lower at 1.18 and 1.23 respectively.

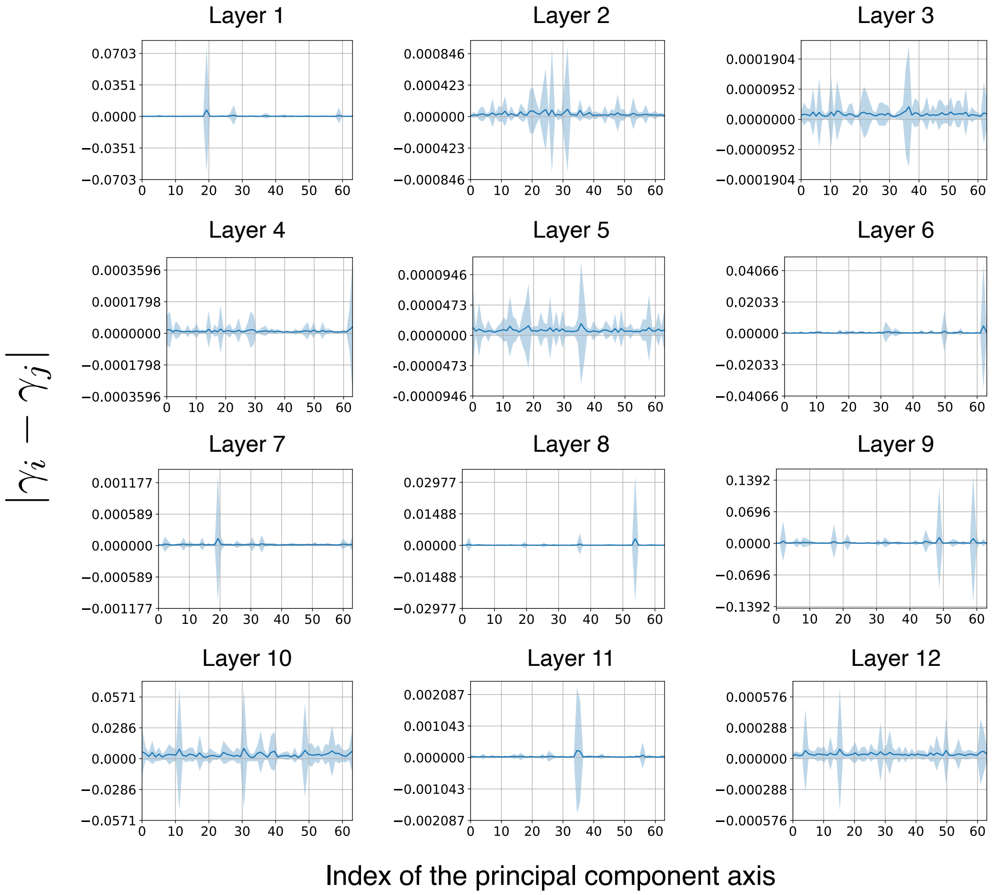

Figure 5: Plot of the mean and standard deviation of the differences in coordinate values of constant vector $\mathbf{1}\lambda_d$ for $d = 1, \dots, D_v$ for all 12 layers of a ViT-tiny model. The mean should be 0 with small standard deviations when $v_{dj} \approx \frac{a_{dj}}{g(\boldsymbol{k}_j)} - \frac{1}{N}\sum_{j'=1}^{N} \frac{a_{dj'}}{g(\boldsymbol{k}_j)}$.

E.2 Pairwise Absolute Differences of γ_i and γ_j

In Figure 5, we provide the plot of the absolute differences of the coordinates of $\frac{\widetilde{\boldsymbol{K}}_{\boldsymbol{\varphi}}\hat{\boldsymbol{a}}_d}{N\hat{\boldsymbol{a}}_d}$ in all 12 layers of a ViT-tiny model as elaborated in Section 2.2.2. This would be a constant vector within our framework and the plots provide empirical evidence to justify our claim. In all of the layers, the means are noticeably close to 0 and the standard deviations are also small suggesting that we indeed recovered a constant vector.

E.3 ImageNet-C Results by Corruption Type

In Table 4, we provide the full results of various RPC-SymViT models against the standard SymViT on ImageNet-C for all corruption types averaged over the 5 severity levels. In all types, except for Zoom Blur, the RPC-SymViT model with 6 iterations in the 1st layer outperforms SymViT.

E.4 Square Attack

Square attack [3] is a score-based black box attack that does not use local gradient information, hence it is not affected by gradient masking. It operates based on a random search scheme that modifies square-shaped regions such that at each iteration, the perturbation lies approximately at the boundary of the feasible set. We evaluate our RPC-SymViT(6iter/layer1) variant on square attacked ImageNet-1K validation set and compare it to the baseline. Our result of the top-1 and top-5 accuracy in Table 5 illustrates that the effectiveness of RPC-Attention against adversarial attacks is not solely due to a form gradient masking as we still significantly outperform the baseline on square.

Table 4: Top-1 accuracy (%) and mean corruption error (mCE) of all RPC-SymViT variants and SymViT on each corruption type in ImageNet-C. RPC-SymViT ($n$iter/layer1) applies n PAP iterations only at the first layer. RPC-SymViT ($n$iter/all-layer) applies n PAP iterations at all layers.

Corruption Type	Model/ Metric	SymViT	RPC-SymViT (2iter/all-layer)	RPC-SymViT (4iter/layer1)	RPC-SymViT (5iter/layer1)	RPC-SymViT (6iter/layer1)
Brightness	Top-1 ↑	63.21	62.97	63.74	64.19	64.31
	mCE ↓	65.16	65.69	64.22	63.43	63.22
Contrast	Top-1 ↑	50.25	49.59	49.30	49.84	50.18
	mCE ↓	58.53	59.08	59.42	58.79	58.39
Defocus Blur	Top-1 ↑	35.47	35.61	35.54	35.26	36.38
	mCE ↓	78.71	78.53	78.62	78.96	77.59
Elastic Transform	Top-1 ↑	44.26	44.52	44.68	44.84	44.72
	mCE ↓	86.28	85.87	85.63	85.38	85.56
Fog	Top-1 ↑	42.72	45.49	46.03	46.59	46.40
	mCE ↓	69.91	66.53	65.87	65.19	65.42
Frost	Top-1 ↑	44.55	45.06	45.93	46.24	46.27
	mCE ↓	67.08	66.46	65.41	65.04	65.01
Gaussian Noise	Top-1 ↑	40.97	41.64	42.22	41.67	42.51
	mCE ↓	66.59	65.84	65.19	65.80	64.85
Glass Blur	Top-1 ↑	27.79	28.12	28.48	27.87	28.05
	mCE ↓	87.39	87.00	86.56	87.30	87.07
Impulse Noise	Top-1 ↑	38.56	38.98	40.24	39.46	40.33
	mCE ↓	66.59	66.14	64.77	65.61	64.67
JPEG Compression	Top-1 ↑	46.28	46.23	47.92	48.41	48.63
	mCE ↓	88.58	88.65	85.87	85.06	84.70
Motion Blur	Top-1 ↑	40.47	40.00	41.72	41.81	42.22
	mCE ↓	75.75	76.43	74.16	74.04	73.51
Pixelate	Top-1 ↑	39.36	39.81	41.13	41.86	42.25
	mCE ↓	84.47	83.84	82.01	81.00	80.45
Shot Noise	Top-1 ↑	38.83	39.20	39.52	39.03	39.71
	mCE ↓	68.38	67.98	67.61	68.16	67.40
Snow	Top-1 ↑	38.27	37.87	38.75	39.12	38.90
	mCE ↓	71.22	71.67	70.66	70.23	70.48
Zoom Blur	Top-1 ↑	30.69	29.49	29.97	30.20	30.48
	mCE ↓	86.82	88.32	87.72	87.43	87.08

Table 5: Top-1 and Top-5 accuracy (%) of RPC-SymViT(6iter/layer1) and baseline SymViT on square attacked ImageNet-1K validation data. RPC-SymViT ($n$iter/layer1) applies n PAP iterations only at the first layer.

Model	Square attack	
	Top-1 ↑	Top-5 ↑
SymViT (baseline)	41.50	79.79
RPC-SymViT (6iter/layer1)	**43.64**	**80.81**

E.5 Results on ADE20K Image Segmentation

We further evaluate the benefits of our method by implementing RPC-Attention in a Segmenter model [72] and providing results on the ADE20K image segmentation task [95] in Table 6. We report the Mean Accuracy and Mean Intersection-Over-Union (IOU) for a Segmenter with a RPC-SymViT backbone and compare it against a baseline SymViT backbone. To assess the robustness of each model, we corrupt the ADE20K dataset with the same 15 corruption types in ImageNet-C and report the same metrics averaged over all corruption types in Table 6 as well. Details of our implementation can be found in Appendix A.3. On both datasets, RPC-SymViT outperforms the baseline substantially.

E.6 Results on RPC-SymViT-base

To show that RPC-Attention is not limited to small-scale models, we conduct experiments on a larger model, SymViT-base with 12 transformer layers, 12 heads per layer, and a hidden dimension of 768. We train a SymViT-base with RPC-Attention in all layers using 2 iterations on ImageNet-1K for

Table 6: Mean accuracy (%) and mean Intersection-Over-Union (IOU) of RPC-SymViT and SymViT on clean ADE20K and corrupted ADE20K dataset. RPC-SymViT ($n$iter/layer1) applies n PAP iterations only at the first layer. RPC-SymViT ($n$iter/all-layer) applies n PAP iterations at all layers.

Model	ADE20K		Corrupted ADE20K	
	Mean Acc. ↑	Mean IOU ↑	Mean Acc. ↑	Mean IOU ↑
SymViT	44.27	34.00	14.85	10.47
RPC-SymViT (4iter/layer1)	**45.61**	**34.69**	16.06	11.47
RPC-SymViT (5iter/layer1)	45.51	34.63	16.15	11.35
RPC-SymViT (6iter/layer1)	45.27	34.24	**16.44**	**11.60**
RPC-SymViT (2iter/all-layer)	43.61	33.5	15.04	10.64

Table 7: Top-1, top-5 accuracy (%) , mean corruption error (mCE), and area under the precision-recall curve (AUPR) of RPC-SymViT-base and SymViT-base on clean ImageNet-1K data and popular standard robustness benchmarks for image classification. RPC-SymViT-base ($n$iter/all-layer) applies n PAP iterations at all layers.

Model	IN-1K		IN-A	IN-C		IN-O
	Top-1 ↑	Top-5 ↑	Top-1 ↑	Top-1 ↑	mCE ↓	AUPR ↑
SymViT-base (baseline)	80.62	94.78	24.03	58.88	52.57	23.96
RPC-SymViT-base (2iter/all-layer)	**80.72**	**94.82**	**24.97**	**59.29**	**52.00**	**26.01**

Table 8: Top-1 and top-5 accuracy (%) of RPC-SymViT-base and SymViT-base on PGD and FGSM attacked ImageNet-1K validation data with the highest perturbation budget. RPC-SymViT-base ($n$iter/all-layer) applies n PAP iterations at all layers.

Model	PGD		FGSM	
	Top-1 ↑	Top-5 ↑	Top-1 ↑	Top-5 ↑
SymViT-base (baseline)	12.11	25.55	53.61	78.05
RPC-SymViT-base (2iter/all-layer)	**13.41**	**26.68**	**54.68**	**78.06**

300 epochs. We refer to this model as RPC-SymViT-base (2iter/all-layer). We compare our RPC-SymViT-base with the baseline on the same standard robustness benchmarks as before, ImageNet-C, ImageNet-A, and ImageNet-O, as well as on white box attacks PGD and FGSM with the highest perturbation budgets. These results are in Table 7 and 8 respectively and show that RPC-Attention is also effective in SymViT-base.

E.7 Results on RPC-Attention in FAN

To validate that our RPC-Attention is also complementary with SOTA transformer models and can be combined with those methods to improve the SOTA results, we have conducted additional experiments in which we incorporate our RPC-Attention with FAN [96]. FAN is one of the top transformer models that obtain SOTA results on accuracy and robustness. A FAN model augments the MLP layer that follows the standard self-attention with a new channel attention (CA) block. This CA computes an attention matrix along the channel dimension, taking advantage of the feature covariance and allowing the model to filter out irrelevant information.

We use the FAN-ViT-tiny (FAN-tiny) variant with a symmetric attention for training. In our RPC-Attention + FAN (RPC-FAN-tiny), we replace the attention blocks in the first layer of FAN with our RPC-Attention that runs 4 PAP iterations with hyperparameter values of $\lambda = 4$ and $\mu = ND/4\|\boldsymbol{K}\|_1$. Both our RPC-FAN-tiny and the baseline FAN-tiny are trained for 300 epochs on the ImageNet-1K object classification task. We summarize our results in Tables 9 and 10. RPC-FAN-tiny outperforms the baseline FAN-tiny on all evaluated benchmarks, including ImageNet-1k, ImageNet-R, and ImageNet-A. Additionally, on PGD and FGSM attacked data, RPC-FAN-tiny significantly outperforms FAN-tiny by over 3%.

E.8 Results on downstream Natural Language Understanding tasks

We conduct additional experiments on several downstream Natural Language Understanding tasks to illustrate the effectiveness of RPC-Attention during fine-tuning as well. The Stanford Sentiment

https://doi.org/10.52202/079017-3215

Table 9: Top-1, top-5 accuracy (%) of RPC-FAN-tiny and FAN-tiny on clean ImageNet-1K data and popular standard robustness benchmarks for image classification. RPC-FAN-tiny ($n$iter/layer1) applies n PAP iterations only at the first layer.

Model	IN-1K		IN-R	IN-A
	Top-1 ↑	Top-5 ↑	Top-1 ↑	Top-1 ↑
FAN-tiny (baseline)	77.89	94.20	41.79	13.40
RPC-FAN-tiny (4iter/layer1)	**77.98**	**94.27**	**42.02**	**13.55**

Table 10: Top-1 and top-5 accuracy (%) of RPC-FAN-tiny and FAN-tiny on PGD and FGSM attacked ImageNet-1K validation data with the highest perturbation budget. RPC-FAN-ViT ($n$iter/layer1) applies n PAP iterations only at the first layer.

Model	PGD		FGSM	
	Top-1 ↑	Top-5 ↑	Top-1 ↑	Top-5 ↑
FAN-tiny (baseline)	2.92	4.86	32.01	61.72
RPC-FAN-tiny (4iter/layer1)	**6.25**	**10.01**	**35.12**	**63.40**

Table 11: Train and validation accuracy (%) of RPC-SymSMoE, SymSMoE, RPC-SMoE (asymmetric) and SMoE (asymmetric) on SST-2 and IMDB downstream tasks when pre-trained on WikiText-103. RPC is only implemented during fine-tuning, not pre-training and runs for 2 iterations for all layers in each model.

Model	SST-2		IMDB	
	Train Acc ↑	Valid Acc ↑	Train Acc ↑	Valid Acc ↑
SymSMoE (baseline)	63.12	69.20	-	-
RPC-SymSMoE	**74.48**	**76.27**	-	-
SMoE (baseline)	53.20	69.38	64.12	65.74
RPC-SMoE	**58.87**	**70.63**	**74.36**	**73.96**

Table 12: Validation and test accuracy of RPC-Attention implemented in a pre-trained transformer language model (RPC-LM) during fine-tuning versus the baseline transformer language model (Baseline-LM). We fine-tune both models on the 5-class Stanford Sentiment Treebank (SST-5) task with 2 iterations for all layers in the RPC model.

Model	Valid Acc ↑	Test Acc ↑
Baseline-LM	46.51	49.23
RPC-LM	**48.68**	**50.36**

Treebank v2 (SST-2) and IMDB Sentiment Analysis (IMDB) tasks are binary classifications where the goal is to determine if sentences have positive or negative sentiments. We use 2 baseline Sparse Mixture of Experts (SMoE) models, one with symmetric attention (SymSMoE) and one with asymmetric attention (SMoE), pre-trained on WikiText-103 without RPC, then fine-tuned with (RPC-SymSMoE/RPC-SMoE) and without RPC-Attention for comparison. Their results can be found in Table 11. We observe that RPC-models outperform the baseline models significantly on these tasks.

On the 5-class sentiment classification task, Stanford Sentiment Treebank (SST-5), we use a pre-trained transformer language model from the NAACL 2019 tutorial on "Transfer Learning in Natural Language Processing" for fine-tuning. Their publicly available code can be found at https://github.com/huggingface/naacl_transfer_learning_tutorial. The objective of the task is to determine if the sentences have negative, somewhat negative, neutral, somewhat positive, or positive sentiments. We implement RPC-Attention during fine-tuning only (RPC-LM) and compare the results with the baseline model (Baseline-LM) on SST-5 in Table 12. As can be seen from the table, our RPC-Attention is applicable to pre-trained language models and performs significantly better than the baseline. Hence, RPC-Attention is highly effective in downstream natural language understanding tasks and versatile in its application.

https://doi.org/10.52202/079017-3215

Table 13: Flop per sample, run time per sample, memory and number of parameters of each RPC-SymViT variant compared to the SymViT baseline. RPC-SymViT ($n$iter/layer1) applies n PAP iterations only at the first layer. RPC-SymViT ($n$iter/all-layer) applies n PAP iterations at all layers.

Model	Flop/Sample	Sec/Sample (Training)	Sec/Sample (Test)	Memory (Training)	Memory (Test)	Parameters
SymViT (baseline)	1.17M	0.079	0.010	1435MB	1181MB	5.2M
RPC-SymViT (4iter/layer1)	1.22M	0.082	0.010	1441MB	1181MB	5.2M
RPC-SymViT (5iter/layer1)	1.23M	0.084	0.010	1443MB	1181MB	5.2M
RPC-SymViT (6iter/layer1)	1.25M	0.085	0.011	1443MB	1181MB	5.2M
RPC-SymViT (2iter/layer)	1.35M	0.092	0.017	1461MB	1181MB	5.2M

Table 14: Top-1, top-5 accuracy (%) and AUPR of an implementation of RPC-Attention on asymmetric attention evaluated on ImagetNet-1K validation set, ImageNet-R, ImageNet-A and ImageNet-O. These results are compared to the standard asymmetric ViT.

Model	IN-1K		IN-R	IN-A	IN-O
	Top-1 ↑	Top-5 ↑	Top-1 ↑	Top-1 ↑	AUPR ↑
ViT	72.11	90.97	32.41	7.65	17.36
RPC-AsymViT (4iter/layer1)	72.34	91.12	32.23	7.75	17.54
RPC-AsymViT (Sym/Asym)	72.34	91.38	32.79	7.23	17.58

E.9 Computational Efficiency

A possible limitation of introducing an iterative scheme into a deep model is a significant increase in computational overhead. We aim to alleviate that concern by providing the number of flops per sample, run time per sample, memory and number of parameters of each RPC-SymViT variant and the SymViT baseline during both training and test time in Table 13. We observe that RPC-Attention is comparable to the baseline softmax attention across all metrics at test time while only slightly less efficient than the baseline in terms of the number of flops, run time per sample, and memory during training.

E.10 Results on Robust Asymmetric Attention

In this section, we report the results of extending the RPC-SymViT model to the asymmetric attention. However, as the PAP Algorithm. 1 is not designed for multiple data matrices, it is not as effective in the asymmetric case. We implement two variations of the algorithm in an asymmetric attention ViT-tiny with 12 layers. In the 4iter/layer1 version, similar to the symmetric attention case, we run 4 iterations of the algorithm only in the 1st layer of the model by replacing Softmax$(\boldsymbol{K} - \boldsymbol{S}_{k+1} - \mu^{-1}\boldsymbol{Y}_k, \boldsymbol{K} - \boldsymbol{S}_{k+1} - \mu^{-1}\boldsymbol{Y}_k)$ with Softmax$(\boldsymbol{Q}, \boldsymbol{K} - \boldsymbol{S}_{k+1} - \mu^{-1}\boldsymbol{Y}_k)$. For the second version, labeled Sym/Asym, we run the algorithm for 4 iterations in a symmetric attention mechanism in the 1st layer, followed by the standard asymmetric attention in layers 2 to 12. We compare these RPC-AsymViT models to the asymmetric softmax attention ViT.

As we can see from Table 14, both variants only improve slightly over the benchmark on most of the corrupted datasets. Such a result confirms our intuition that the ADMM algorithm does not extend easily to multiple corrupted matrices as it only solves an objective function involving a single low-rank data matrix and its sparse corruption matrix.

F Further Discussion on Related Works

[15] provides a new perspective by emphasizing the asymmetry of the softmax kernel and recovers the self-attention mechanism from an asymmetric Kernel Singular Value Decomposition (KSVD) using the duality of the optimization problem. Separately, our kernel PCA perspective derives softmax attention and addresses the asymmetry of attention through a projection of the query vectors in feature space. While there are similarities between KSVD and kernel PCA, in the sense of finding low rank approximations and low dimensional representations of the data, the primal objective functions are different and we do not need to consider the dual form. Another related work views transformers from the perspective of Support Vector Machines [76]. Though, kernel PCA can be formulated in a similar fashion as a least squares SVM problem as explained in [74], our work focuses on the forward

https://doi.org/10.52202/079017-3215

pass of attention and show that it can recover a kernel PCA solution. In contrast, [74] examines the backward pass and optimization geometry of an attention layer towards a hard margin SVM solution that separates optimal tokens from non-optimal ones, under certain assumptions of the loss function, initial conditions, and certain SVM constraints. Furthermore, using our framework, we are able to predict the exact explicit form of the value matrix in self-attention, demonstrating that this matrix captures the eigenvectors of the Gram matrix of the key vectors in a feature space. To the best of our knowledge, our work is the first to show this insight.

Hamburger in [25] models the global context discovery as the low-rank recovery of the input tensor and solves it via matrix decomposition. Both Hamburger and our Attention with Robust Principal Components (RPC-Attention) try to recover clean signal subspaces via computing a low-rank approximation of a given matrix. The key differences between our RPC-Attention and Hamburger are: (1) Our RPC-Attention finds a low-rank approximation of the key matrix while Hamburger finds a low-rank approximation of the input matrix, and (2) Our RPC-Attention models the corruption by a sparse matrix while Hamburger does not enforce this condition. The entries of this sparse corruption can have an arbitrarily large magnitude and help model grossly corrupted observations in which only a portion of the observation vector is contaminated by gross error. Numerous critical applications exist where the data being examined can be naturally represented as a combination of a low-rank matrix and a sparse contribution, such as video surveillance, face recognition, and collaborative filtering [11].

[89] derives each component in a Graph Neural Network (GNN) from the unfolded iterations of robust descent algorithms applied to minimizing a principled graph regularized energy function. In particular, propagation layers and nonlinear activations implement proximal gradient updates, and graph attention results from iterative reweighted least squares (IRLS). While this is an interesting approach, it has not been extended to explaining the architecture of transformers, including self-attention, yet. In contrast, our kernel principal component analysis (kernel PCA) allows us to derive self-attention in transformers, showing that the attention outputs are projections of the query vectors onto the principal components axes of the key matrix in a feature space.

[2] and [6] implement each layer as an optimization and fixed-point solver, respectively. In particular, an OptNet layer in [2] solves a quadratic program, and a Deep Equilibrium layer in [6] computes the fixed point of a nonlinear transformation. Different from these layers, our RPC-Attention solves a Principal Component Pursuit - a convex program. Also, both OptNet layer in [2] and Deep Equilibrium layer in [6] do not shed light on the derivation and formulation of self-attention, which our kernel PCA framework does.

G Broader Impacts

Our research improves both clean data processing and robust performance, especially in areas with significant social relevance. Specifically, we demonstrate enhanced results in image segmentation, benefiting self-driving cars, and language modeling, enhancing AI chatbot assistants. We show notable improvements in resisting adversarial attacks, aiming to protect crucial AI systems from malicious actors. Additionally, we achieve competitive performance in language modeling with contaminated data, which is often encountered in real-world situations. Despite the potential for AI misuse, our research presents substantial advancements in fundamental architectures and theory, which we hope will inspire further socially beneficial developments.

https://doi.org/10.52202/079017-3215

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction are clearly stated in the **Contribution** in the Introduction. These claims accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in the Conclusion.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

https://doi.org/10.52202/079017-3215

Answer: [Yes]

Justification: All theoretical results in the paper are given together with the full set of assumptions and complete/correct proofs (See Section 2.1 in our manuscript).

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide the experiment details in the Experiment Details Section in the Appendix of our manuscript. We also provide the source code so that the results in the paper can be easily reproduced.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

https://doi.org/10.52202/079017-3215

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide the source code so that the results in the paper can be easily reproduced. We verify our proposed methods using public benchmarks (See the Experimental Results Section in our manuscript)

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify all the training and test details necessary to understand the results in the Experiment Details Section in the Appendix of our manuscript.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report error bars suitably and correctly defined of the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

https://doi.org/10.52202/079017-3215

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide sufficient information on the computer resources for all experiments in our Experimental Results Section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss broader impacts in Appendix G.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

https://doi.org/10.52202/079017-3215

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the githubs we use and the baselines we compare with in our manuscript.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

https://doi.org/10.52202/079017-3215

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

https://doi.org/10.52202/079017-3215

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-3215