ATP: Enabling Fast LLM Serving via Attention on Top Principal Keys

Anonymous ACL submission

Abstract

We propose a new attention mechanism with linear complexity, ATP, that fixates **A**ttention on **T**op **P**rincipal keys, rather than on each individual token. Particularly, ATP is driven by an important observation that input sequences are typically low-rank, i.e., input sequences can be represented by a few principal bases. Therefore, instead of directly iterating over all the input tokens, ATP transforms inputs into an orthogonal space and computes attention only on the top principal bases (keys). Owing to the observed low-rank structure in input sequences, ATP is able to capture semantic relationships in input sequences with a few principal keys. Furthermore, the attention complexity is reduced from *quadratic* to *linear* without incurring a noticeable performance drop. ATP further reduces complexity for other linear layers with low-rank inputs, leading to more speedup compared to prior works that solely target the attention module. Our evaluations on various models (e.g., BERT and Llama) demonstrate that ATP achieves comparable accuracy with much lower computation and memory complexity than the standard attention mechanism. In particular, ATP barely loses accuracy with only $1/2$ principal keys, and only incurs around 2% accuracy drops with $1/4$ principal keys.

1 Introduction

Transformers with self-attention have become a mainstream model architecture in many machine-learning tasks on natural language processing (Wolf et al., 2020), and computer vision (Khan et al., 2022; Dosovitskiy et al., 2020). In particular, owing to the attention mechanism, transformers have been demonstrated to be more effective in learning semantic relationships from input sequences. This drives transformers to become the backbone of current large language models (LLMs) like ChatGPT (OpenAI) and Copilot (Microsoft).

Despite their remarkable utility in real-world applications, transformers with standard self-attention, however, incur *quadratic* complexity in terms of sequence length (Vaswani et al., 2017). To be specific, considering an input sequence with length L (i.e., L tokens), each attention layer needs $\mathcal{O}(L^2)$ computation and memory complexity on attention operations. Such a quadratic degree of complexity renders transformers difficult to scale with long input sequences. As a result, most LLM services at scale backed by transformers incur significant computation and memory footprints, which can only be afforded by large companies with sufficient computing power (Samsi et al., 2023). To meet memory and computation resource constraints during deployment, some transformer models (Devlin et al., 2018; Lan et al., 2020; Radford et al., 2018; Touvron et al., 2023) usually come with a hard constraint on sequence length. However, in many real-world tasks such as question-answering (Wang et al., 2019), text summarization (El-Kassas et al., 2021), enabling long sequence length is crucial for capturing semantic relationships in a broader context, and improving models' performance.

Therefore, at the core of transformers and LLM services, lightweight self-attention mechanisms play a key role in improving model performance with longer sequences, as well as computation and memory efficiency in deployment.

Current works on reducing the complexities of transformers can be categorized in two ways. The first line of research usually exploits redundancy in query/key/value matrices or attention maps, while the second approximates the `Softmax`-based attention with linear complexity.

Along the first line of works, Vyas et al. (2020) reduces attention complexity via clustering queries and only computes attention output for each cluster. Its performance hinges on the performance of clustering as well as the dimension in queries. On the other hand, Linformer (Wang et al., 2020)

chooses to reduce the number of keys/values via a low-dimensional projection. A pre-defined or learnable projection layer is inserted into each attention layer. However, such a projection layer lacks a rigorous guarantee to preserve information in inputs. Compared to simply approximating queries, keys, or values, another line of work approximates the Softmax-based attention using randomized feature mapping(Rahimi and Recht, 2007). In these works, standard attention is regarded as a kernel method, and can be approximated with low-dimensional kernels. For instance, Performers (Choromanski et al., 2020) shows that Softmax attention can be converted to a Gaussian kernel function. Therefore, self-attention can be potentially calculated at a lower dimension with linear complexity, as done in a Gaussian kernel function. Following such an idea, several works explore different kernels to approximate Softmax attention (Katharopoulos et al., 2020a). However, the approximation methods with randomized feature mapping need to trade off approximation accuracy and the number of random features. A low approximation error needs more random features but increases approximation complexity (Choromanski et al., 2020). Furthermore, while the aforementioned works reduce the complexity of the attention mechanism to linear, they still have not been seen in large-scale language models such as Llama (Touvron et al., 2023). One reason is that these methods cannot effectively preserve information when performing attention in low dimensions.

In this paper, we propose a new attention mechanism with linear complexity, that maintains model performance with significantly reduced computation and memory costs. The new attention mechanism, called ATP, is the first work that adapts self-attention with a low-rank structure in input embeddings. In particular, ATP first analyzes an input sequence's structure and obtains orthogonal bases. We observe that input sequences usually exhibit a high correlation among tokens, with a few orthogonal bases being more important than the rest. Then ATP computes attention only on the top principal bases (we call them *principal keys*), rather than iterating over all keys as in the standard attention layer. Owing to the low-rank structure in input sequences, the new self-attention mechanism with few principal keys/values is sufficient to capture semantic relationships among tokens. As a result, compute and memory complexity is reduced from quadratic to linear in terms of sequence length.

Furthermore, by exploiting low-rank structure in inputs, not only is the complexity of attention reduced, but also the complexity of other linear layers. Hence, ATP achieves further computation reductions compared to prior works focusing solely on the Softmax operation.

Our evaluations on various models (e.g., BERT and Llama) demonstrate ATP still maintains comparable accuracy with small fractional principal keys/values. In particular, with only $1/4$ principal keys, ATP achieves accuracy almost as the original model. With only $1/4$ principal keys, ATP only incurs around 2% accuracy drop on BERT-base and Llama2 models.

2 Preliminaries and Related Works

2.1 Standard Self-Attention

Standard self-attention consists of three matrices: queries Q, keys K, and values $V \in \mathbb{R}^{L\times d'}$, where L is sequence length and d' is the hidden dimension. For each query vector $\boldsymbol{q} \in Q$, the self-attention applies dot-product with all keys, followed by a Softmax op to compute a score on each key. Each score denotes a weight on the corresponding values. Then the attention output $A(\boldsymbol{q})$ is obtained as a weighted average of all values:

$$A(\boldsymbol{q}) = \texttt{Softmax}(\boldsymbol{q} \cdot K^T/\sqrt{d}) \cdot V. \quad (1)$$

The query/key/value matrices are obtained by projecting input $X \in \mathbb{R}^{L\times d}$ with parameter $W^Q, W^K, W^V \in \mathbb{R}^{d\times d'}$ as

$$Q, K, V = X \cdot \{W^Q, W^K, W^V\}. \quad (2)$$

In essence, the self-attention mechanism finds the relations between a query and all keys, which are measured by probability after Softmax. Then, it averages corresponding values with the notion that a key closer to the query should be assigned larger weights (i.e., probability after Softmax).

2.2 Related Works on Efficient Self-Attention

Given an input sequence with length L, the standard self-attention needs to perform L^2 dot-products for all token vectors to get the whole attention map. As a result, it incurs complexity of $\mathcal{O}(L^2)$ on computations and memory, which makes it difficult to scale with long inputs in many tasks. As a result, current LLM services usually require a significant amount of memory and computing power, in order to support long sequences.

In some cases, facing actual resource constraints, some LLMs may need to limit the sequence length.

Current literature typically mitigates the limitation via exploiting sparsity or redundancy in attention matrices (Roy et al., 2021; Sun et al., 2021; Vyas et al., 2020; Katharopoulos et al., 2020a; Wang et al., 2020), or approximating the self-attention operation. For sparsity and redundancy in attention, exploiting low-rank structures in query/key/value and attention maps shows great potential. For instance, by exploring redundancy in input query vectors, Vyas et al. (2020) propose to first cluster query vectors, and use *cluster centroid vectors* to represent all query vectors of the same cluster. Hence, for all queries in the same cluster, it only needs to compute the attention score once on the centroid vector. With a reduced number of vectors when performing self-attention, it reduces the complexity of self-attention from quadratic to linear. However, the cost is a noticeable error by approximating many queries with the same cluster, thereby leading to performance degradation. On the other hand, Wang et al. (2020) project key and value matrices into a low-dimensional space. Specifically, with r keys and values in the low-dimensional space, the method only needs to perform an attention op on r keys rather than L keys as in the standard self-attention mechanism. However, due to the fact that the projection matrix is pre-defined and learned from scratch, it is not guaranteed that the low-dimensional projection is effective in preserving information in the original key and value matrices (Wang et al., 2020). Besides query/key/value matrices, Nguyen et al. (2021); Han et al. (2023) directly exploit redundancy in the attention map, and approximate the attention map with low-rank and sparse matrices. Therefore, computation and memory costs of self-attention can also be reduced.

Besides removing redundancy in self-attention operations, current works along the second line of research attack the problem via approximating `Softmax` operations with kernelization. Typically, Choromanski et al. (2020) regard self-attention as `Softmax` kernels: $\texttt{exp}(\boldsymbol{q} \cdot \boldsymbol{k}^T)$ with query $\boldsymbol{q}$ and key $\boldsymbol{k}$, and approximate it with the Gaussian kernel function(Rahimi and Recht, 2007). Specifically, it estimates `Softmax` as: $\texttt{exp}(\boldsymbol{q} \cdot \boldsymbol{k}^T) \rightarrow \mathbb{E}\left[\phi(\boldsymbol{q}) \cdot \phi(\boldsymbol{k})^T\right]$, where kernel function $\phi(\cdot)$ maps a vector to a low-dimensional space. Therefore, the dimension after kernelization is reduced, leading to a reduction in self-attention operations. Along this line, other works (Katharopoulos et al., 2020b; Nguyen et al., 2021) explore different kernel functions to approximate the self-attention function. While the complexity is reduced, these kernel-based approximations still incur large `Softmax` approximation errors given large hidden dimensions in large models.

Therefore, *a lightweight self-attention mechanism with linear complexity is still needed, especially for current large models with huge computation and memory footprints.*

3 Lowrank Structure in Sequences

Low-rank structures in inputs of language models are an essential component, that, surprisingly, is rarely exploited in current models for better computation and memory efficiency. Compared to model parameters (Hu et al., 2021), inputs and internal hidden states are usually more correlated, which can be potentially exploited. Such a property has also been observed in vision problems (Niu et al., 2022; Andriushchenko et al., 2023) and used to reduce the complexity of convolution operations. This paper is the first work that investigates the low-rank structure of input sequences in language models, and, importantly, its potential to computation and memory saving. In this section, we first analyze low-rank structures in transformers' input sequence. Then, in the next section, we present `ATP` that leverages low-rank structures in inputs and performs self-attention with significantly reduced computation and memory footprints.

Transformer models comprise a stack of self-attention layers. Each self-attention layer takes input state $X \in \mathbb{R}^{L \times d}$, and computes output state $Y \in \mathbb{R}^{L \times d}$, where L denotes the sequence length, d is the dimension of each hidden state vector. Each state vector corresponds to a token in the input sequence. Owning to the semantic relationships among tokens, these vectors are also correlated. To formally measure such correlations, we adopt a metric called `SVD-Entropy`.

In detail, we apply singular value decomposition (`SVD`) to the hidden state as

$$X \quad \xrightarrow{\text{SVD}} \quad \sum_{i=1}^{L} \sigma_i \cdot \boldsymbol{u}_i \cdot \boldsymbol{v}_i^T. \tag{3}$$

We assume $L \leq d$ without loss of generality. With Eq(3), we attains singular values $\{\sigma_i\}$ and corresponding principal components $\{\boldsymbol{v}_i\}$.

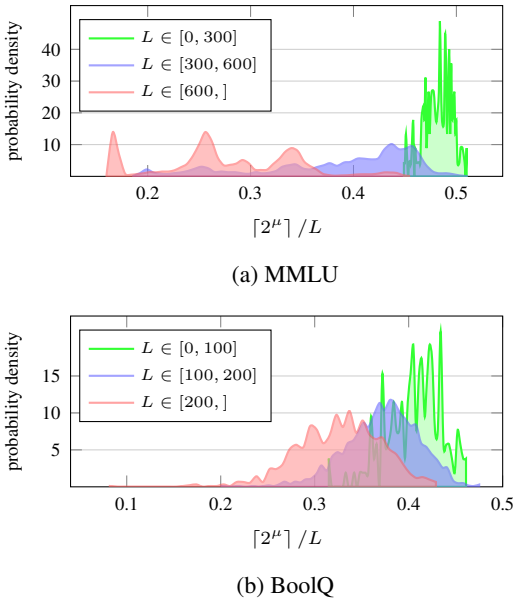
Then, based on Niu et al. (2022), we compute SVD-Entropy as the "low-rankness" of X,

$$\mu = -\log\left(\sum_{i=1}^{L} \bar{\sigma}_i^2\right), \tag{4}$$

where $\bar{\sigma}_i = \frac{\sigma_i}{\sum_{i'=1}^{L} \sigma_{i'}}$.

According to Niu et al. (2022), $\lceil 2^\mu \rceil$ can denote the number of necessary principal components to sufficiently approximate input X. $\lceil 2^\mu \rceil \ll L$ implies that input state vectors in X are highly correlated such that only a few principal components are sufficient to represent X.

With such a measure, we analyze the low-rank structure of hidden states in language models. Figure 1 shows the distribution of low-rankness after Llama-2's embedding layer on BoolQ and MMLU datasets, measured by ratio $\lceil 2^\mu \rceil / L$. A small ratio implies that the embedding of a sequence is more low-rank. We can easily observe that embeddings of all sequences are highly low-rank, where 50% or even fewer principal components are sufficient to approximate embedding vectors without error. Moreover, longer sequences usually exhibit more low-rank structure compared to shorter sequences. Note that the observation implies that exploiting the low-rankness of input data can be more effective compared to the low-rankness of models (Hu et al., 2021). Such a crucial observation presents great potential for reducing the dimension of inputs, thereby leading to more efficient self-attention with reduced computation and memory complexities, especially for long sequences. Low-rankness analysis of other models is deferred to Appendix A.



(a) MMLU

(b) BoolQ

Figure 1: Distribution of low-rankness of Llama-2's embedding on MMLU and BoolQ dataset, measured by ratio $\lceil 2^\mu \rceil / L$. Almost all sequences can be sufficiently approximated with less than half principal components without incurring error. Longer sequences exhibit a more low-rank structure.

4 ATP Methodology

In this section, we introduce ATP, a generic transformer architecture with a new efficient self-attention. ATP introduces a rank-aware self-attention mechanism that reduces the complexity of self-attention to linear given the low-rank structure in input sequence embeddings.

4.1 Self-Attention with Low-Rank Inputs

Given low-rank input $X \in \mathbb{R}^{L \times d}$ with r principal components, we write it as

$$X = U \cdot X', \tag{5}$$

where $U \in \mathbb{R}^{L \times r}$, and $X' \in \mathbb{R}^{r \times d}$ denotes the principal components. Since X is low-rank, query/key/values matrices obtained by projecting X are also low-rank. That is,

$$\begin{aligned} Q, K, V &= U \cdot X' \cdot \{W^Q, W^K, W^V\} \\ &= U \cdot \{Q', K', V'\}. \end{aligned} \tag{6}$$

By the matrix rank inequality (Banerjee and Roy, 2014), we have $\texttt{rank}(\{Q, K, V\}) \leq \texttt{rank}(X') = r$.

Then we start from the standard self-attention, and show the computations can be significantly reduced with low-rank keys/values. We omit the normalization in Softmax and write self-attention with query $\boldsymbol{q}$ on all keys/values as $\texttt{exp}(\boldsymbol{q}, K^T) \cdot V$. With low-rankness of input X, we can break down the self-attention as

$$\texttt{exp}(\boldsymbol{q} \cdot K^T) \cdot V = \texttt{exp}(\boldsymbol{q} \cdot K'^T \cdot U^T) \cdot U \cdot V' \tag{7}$$

By the Taylor expansion on the exp function on ech value, we have the following approximation,

$$\begin{aligned} &\texttt{exp}(\boldsymbol{q}, K^T) \cdot V \\ &\simeq \mathbf{1} \cdot U \cdot V' + \boldsymbol{q} \cdot K'^T \cdot U^T \cdot U \cdot V' \\ &= \mathbf{1} \cdot U \cdot V' + \boldsymbol{q} \cdot K'^T \cdot V' \\ &= (\mathbf{1} \cdot U + \boldsymbol{q} \cdot K'^T) \cdot V' = A' \cdot V', \end{aligned} \tag{8}$$

where $\mathbf{1} \in \mathbb{R}^{1\times L}$, and $U^T \cdot U = I$. Similar as Softmax, normalization is applied row-wise on the new attention map A'.

Eq(8) shows that self-attention on all token vectors X can be converted to attention on all principal keys K'. More importantly, different from the standard self-attention where each key corresponds to a token in the input sequence, these principal keys denote all principal bases drawn from X'. That is, ATP *converts the attention operation from individual token vectors to principal basis vectors.* The observation is very crucial given low-rank input X. The reason is that, given low-rank input with $r \ll L$, based on Eq(8), for each query, we only need to perform dot-product on r vectors, rather than L vectors as in the standard self-attention. Therefore, the self-attention does not incur $\mathcal{O}(L^2)$ computation and memory costs. Instead, the costs scale linearly with sequence length, L, and the number of principal components, r. Figure 2 shows a point-to-point comparison between the standard self-attention and the low-rank self-attention. The low-rank self-attention shares a similar procedure as the stand self-attention, while the difference is that the low-rank self-attention performs dot-product on r principal keys.

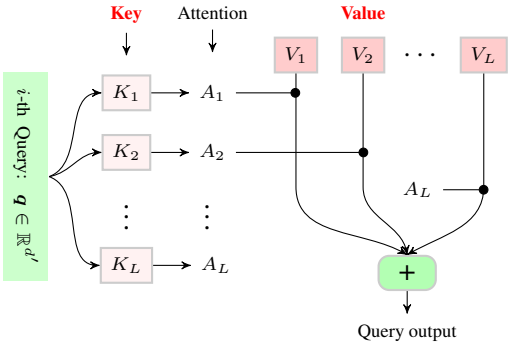
Remark 4.1 *Unlike works such as (Wang et al., 2020) that attain low-dimensional key/value matrices via hard-coded/learnable projection, we adopt a more rigorous method based on* SVD *to find the optimization low-dimensional space, that preserves most energy of input X with r principal components.*

4.2 Tansformers with Low-Rank Attention

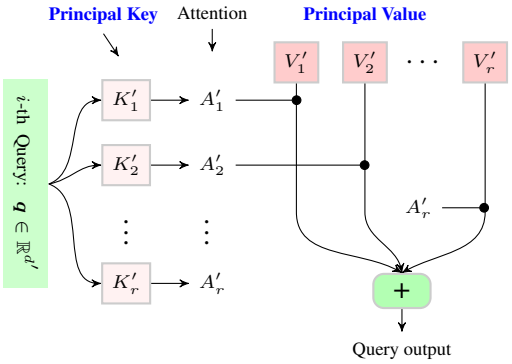
With the low-rank self-attention above, we can adapt the transformer architecture to input sequences with highly low-rank structure. To the best of our knowledge, this is the first adaptation that takes input low-rank structure into model design, and reduces complexities for the whole pipeline.

As a transformer model is usually built with a stack of encoder/decoder layers with the same architecture, to simplify, we only show the architecture adaptation for one encoder/decoder layer, which will be replicated to the rest of the layers.

The first step is to analyze input X and attain its principal components. To that end, we decompse input X using SVD, and attain the principal components X' based on Eq(3): $X' = [\sigma_1 \boldsymbol{v}_1, \cdots, \sigma_r \boldsymbol{v}_r]$. However, the exact SVD incurs a complexity of



(a) Standard self-attention.



(b) Low-rank self-attention.

Figure 2: Standard self-attention and low-rank self-attention. Low-rank self-attention share the same procedure as the standard self-attention, but with only r principal keys and values.

$\mathcal{O}(Ld^2)$, which can be a performance bottleneck given the large dimension of each vector in X. To avoid such a quadratic complexity, we adopt an approximated SVD algorithm as

$$X' = [\sigma_1 \boldsymbol{v}_1, \cdots, \sigma_r \boldsymbol{v}_r] = \texttt{argmin}_{\sigma, \boldsymbol{u}, \boldsymbol{v}_{1,\cdot,r}} \left\| X - \sum_{i=1}^{r} \sigma_i \cdot \boldsymbol{u}_i \cdot \boldsymbol{v}_i^T \right\|. \tag{9}$$

Essentially, the optimization above is to find r principal components that preserve most energy in X, while ignoring the orthogonality constraint on the components. To simplify the optimization above, σ_i can be fused with $\boldsymbol{v}_i$ to reduce the number of variables to be optimized. By the alternating optimization in Alg 1 in Niu et al. (2022) (duplicated in Appendix B), we can attain r most principal components which preserved most energy in X. Compared to the standard SVD decomposition, the approximation in Eq(9) incurs a linear complexity of $\mathcal{O}(rLd)$, thereby preventing SVD being the

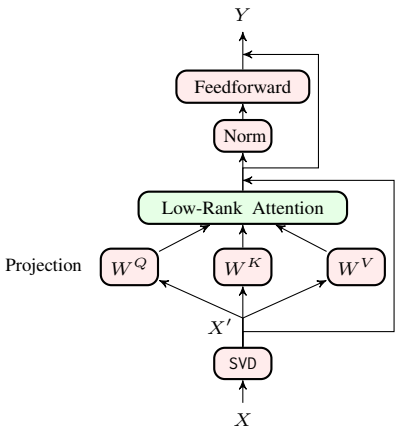


Figure 3: Transformer encoder/decoder with low-rank self-attention. Input X is first fed to SVD to attain the principal components, X'. Then, X' is fed to an encoder/decoder layer with low-rank self-attention.

bottleneck in the whole pipeline.

Then, principal components X' are fed into a self-attention layer to attain principal keys and values as in Eq(6). With the principal keys and values, ATP performs attention as in Eq(8), and feedforward to obtain output states Y. The next encoder/decoder layer follows the same procedure first to attain principal components of Y and perform low-rank self-attention.

Combine with Position Encoding. For absolution or relative position encoding vectors P for a sequence (Devlin et al., 2018; Lan et al., 2020; Shaw et al., 2018), they are added to token embeddings before an encoder/decoder layer. Therefore, we can still directly apply SVD to the input vectors, $X + P$, and obtain principal components for low-rank self-attention.

For rotatory position embedding (Su et al., 2021; Touvron et al., 2023), the position encoding vectors are added after query/key projection. That is, $\boldsymbol{k} = \boldsymbol{x} \cdot W^K \cdot R_i$, where R_i is a rotary matrix corresponding to a transformation for a token at position i, $\boldsymbol{x}$ denotes one input vector in X. While the low-rank structure might change during the rotation, we can still attain a low-rank key matrix by projecting the key matrix into a low-dimension space with U as in Eq(8).

Therefore, the low-rank self-attention mechanism is compatible with current position encoding methods.

4.3 Complexity Analysis

Self-attention with low-rank inputs not only relieves computation and memory pressure for attention operations, but also reduces complexity for other linear layers. Table 1 lists computations and the corresponding complexity of the standard and low-rank self-attention. Due to the reduced number of components in X', query/key/value projection only needs to project r vectors rather than L token vectors as the standard self-attention, thereby resulting in r keys and value vectors with dimension d'. Hence, both the computation and memory during the projection are reduced by L/r. On the other hand, when performing attention, the low-rank attention only needs to compute the attention score on r principal keys, rather than L token keys as the standard self-attention. Therefore, the computation and memory complexities are also reduced by L/r. Note that the additional SVD only incurs computation complexity of $\mathcal{O}(rLd)$, which is linear in term of L, and is relatively small compared to computations in the standard self-attention. In addition, more computation saving can be achieved by decomposing hidden state vectors to the FeedForward layer. In this paper, we mainly focus on self-attention layers.

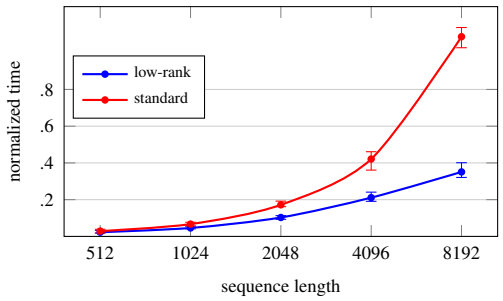


Figure 4: Actual running time of low-rank self-attention compared to the standard mechanism with different sequence lengths (r=128). The running time of the standard self-attention increases quadratically with the sequence length. Low-rank self-attention reduces the running time to almost linear.

Figure 4 shows actual speedups of low-rank self-attention compared to the standard self-attention given different sequence lengths. Note that the standard self-attention, as expected, incurs quadratic running time with increasing input sequence length. On the other hand, the running time of the low-rank self-attention scales almost linearly with sequence length. The time gap between them grows rapidly with long sequences. This shows that the standard self-attention indeed comes with a severe bottleneck on real performance with long sequences, while the low-rank self-attention significantly reduces actual running time.

Mechanism	Standard			Low-rank		
	Computation	Complexity	Memory	Computation	Complexity	Memory
Projection	$X \cdot W$	$\mathcal{O}(Ldd')$	$\mathcal{O}(Ld')$	$X' \cdot W$	$\mathcal{O}(rdd')$	$\mathcal{O}(rd')$
Attention	$Q \cdot K^T$	$\mathcal{O}(L^2d')$	$\mathcal{O}(L^2)$	QK'^T	$\mathcal{O}(rLd')$	$\mathcal{O}(rL)$

Table 1: Computation and memory complexity with low-rank input. Low-rank self-attention reduces the complexity of attention from quadratic to linear. It also reduces complexities for other linear layers (L: sequence length, r: rank, d: dimension of X, d': dimension of hidden state).

5 Empirical Evaluation

In this section, we evaluate the low-rank attention on benchmark models and datasets. To investigate the applicability of low-rank attention in a wide range of applications, we choose models with different sizes. For datasets, we focus on long sequences, which usually incur significant computation and memory pressure during inference.

Model. We choose BERT-base (encoders only) as the small model (Devlin et al., 2018), Llama2-7B (decoder only) as the medium model, and Llama2-13B as the large model (Touvron et al., 2023). Table 2 lists their detailed architecture parameters. Note that all three models adopt the standard self-attention mechanism.

	BERT	Llama2-7B	Llama2-13B
# att layers	12	32	40
# heads/layer	12	32	40
# head dim	64	128	128

Table 2: Architecture parameters of BERT-base, Llama2-7b and Llama2-13B.

Datasets. For BERT-base, we choose SST-2, Squad (Wang et al., 2019), and IMDB(Maas et al., 2011). In particular, the IMDB dataset consists of long sequences that exhibit more low-rank structures. For Llama2-7B and Llama2-13B, we choose two of the official benchmark datasets: MMLU (Hendrycks et al., 2021) and BoolQ (Clark et al., 2019).

5.1 BERT-base

For all datasets, we start from a pre-trained model, replace each self-attention layer with the low-rank self-attention, and finetune the model. Owing to the model size, we finetune full parameters. Training details are provided in Appendix C. Table 3 lists the final model accuracy on SST-2, Squad, and IMDB. We can observe that BERT-base with low-rank self-attention preserves models' performance. In particular, with $1/2$ principal keys used, the model with low-rank self-attention barely loses accuracy. This indicates that owing to the low-rank structure in sequences, $1/2$ principal keys preserve most information in inputs. Surprisingly, we can further see that even only keeping $1/8$ principal keys, the model still achieves a comparable accuracy as the model with standard self-attention.

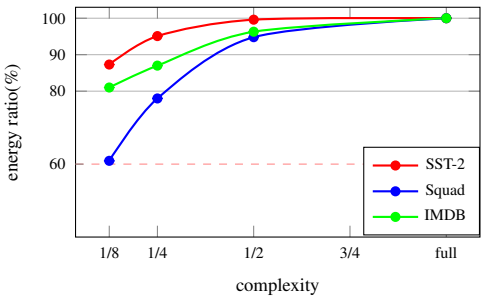


Figure 5: Energy ratio ($\|X'\|_F^2 \,/\, \|X\|_F^2$) in low-rank hidden representations. Embeddings of all three datasets exhibit highly low-rank structures, with $1/2$ principal components preserving almost all energy.

Figure 5 shows the relative energy kept in the low-rank keys. We observe that for $1/2$ principal keys are sufficient to keep almost all energy in inputs, which is aligned with model accuracy in Table 3. On the other hand, compared to Squad and IMDB, SST-2 exhibits a more low-rank structure, with even $1/8$ principal keys still preserving near 90% energy. The observation explains BERT-base's performance on SST-2 that even low-rank self-attention with only $1/8$ principal keys only incurs a $\sim 3\%$ accuracy drop.

5.2 Llama2

We obtain pre-trained Llama2-7B/13B models from the Hugging Face repo [1]. Starting from the pre-trained models, we replace their attention layers with low-rank self-attention. For MMLU and BoolQ, since they have different formats, we will first finetune the model on the datasets for a few

[1] https://huggingface.co/meta-llama

Model	Original	1/2	1/4	1/8
SST-2	92.32 ± 0.2	92.1 ± 0.17	91.0 ± 0.23	89.2 ± 0.26
Squad	88.15 ± 0.3	87.93 ± 0.2	87.23 ± 0.34	84.94 ± 0.28
IMDB	91.45 ± 0.2	90.97 ± 0.19	89.65 ± 0.3	87.28 ± 0.3

Table 3: BERT-base accuracy on SST-2, Squad, and IMDB using low-rank self-attention.

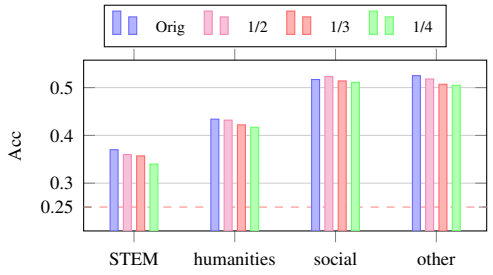
iterations (See Appendix C for more finetuning parameters), and then evaluate their performance on the validation dataset. Appendix D provides prompt formats for MMLU and BoolQ during training and validation. To reduce training workload, we use LoRA (Hu et al., 2021) to finetune the projection matrix for queries/keys/values with rank of 32, and fix other layers.

For MMLU, we obtain the first predicted logit vector from the model given an input sequence, and compute the probability on the four tokens: *A, B, C, D*. The token with the highest probability will be the predicted answer. For BoolQ, we adopt a similar procedure but compute the probability on the two tokens: *Yes, No*, and output the token with the highest probability. Note that we ignore other tokens that might have the highest probability.

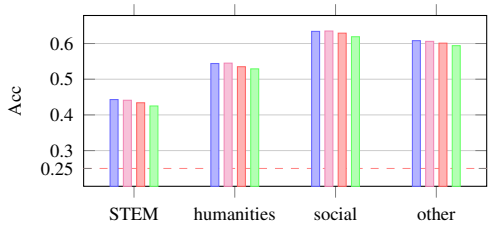
Figure 6 shows the accuracy of Llama2-7B and 13B on MMLU using ATP. We can observe that on all categories, ATP achieves accuracy close to original Llama2-7B and 13B with standard self-attention. In particular, owing to the highly low-rank structure in input sequences, with $1/2$ principal keys, the model performance with ATP is almost identical to the original model. Furthermore, even with only $1/4$ principal keys, ATP still does not incur a significant accuracy drop. Similar performance of LLama2-7B and 13B with the low-rank self-attention holds on the BoolQ dataset, as listed in Table 4. Therefore, ATP effectively leverages low-rank structure in input sequences and performs self-attention with a few top principal keys, leading to performance close to the original model but with significantly reduced complexities.

Model	Orig	1/2	1/3	1/4
7B	0.795	0.791	0.789	0.763
13B	0.839	0.836	0.819	0.816

Table 4: Llama2 on BoolQ with low-rank self-attention. Performance is not greatly affected even a small fraction of principal keys/values are used in attention layers.



(a) Llama2-7B.



(b) Llama2-13B.

Figure 6: LLama2 on MMLU (random guess: 0.25). Low-rank self-attention effectively preserves performance on all subjects, even with $1/4$ principal keys.

6 Conclusion

In this work, we propose a low-rank self-attention mechanism, ATP, significantly reducing computation and memory complexity for transformers and LLMs. ATP leverages low-rank structures in input sequences and sufficiently represents each input sequence with a few top principal components. Then, ATP designs a low-rank self-attention layer that first attains principal keys/values given a low-rank input. Then, it performs attention only on top principal keys/values, rather than on each individual token embedding. Therefore, ATP reduces the attention complexity from quadratic to linear in terms of sequence length. Owing to low-rank structures in input sequences, a few top principal keys/values are sufficient to preserve information in input sequences. Evaluation of BERT and Llama models shows ATP achieves performance close to original models with much-reduced computation and memory footprints.

7 Limitations

Limitations. One of the limitations of this work is that we evaluate ATP on BERT and Llama2 models. While performance on other models may differ. We will evaluate more models and datasets in future works.

Potential Risk. While this work is aimed at lowering the barrier of deploying LLMs, it may be misused by malicious parties to quickly deploy and run adverse LLM services for their purposes.

References

Maksym Andriushchenko, Dara Bahri, Hossein Mobahi, and Nicolas Flammarion. 2023. Sharpness-aware minimization leads to low-rank features. *arXiv preprint arXiv:2305.16292*.

Sudipto Banerjee and Anindya Roy. 2014. *Linear algebra and matrix analysis for statistics*. Crc Press.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. In *International Conference on Learning Representations*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Wafaa S El-Kassas, Cherif R Salama, Ahmed A Rafea, and Hoda K Mohamed. 2021. Automatic text summarization: A comprehensive survey. *Expert systems with applications*, 165:113679.

Insu Han, Rajesh Jarayam, Amin Karbasi, Vahab Mirrokni, David P. Woodruff, and Amir Zandieh. 2023. Hyperattention: Long-context attention in near-linear time.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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

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

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

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Microsoft. Microsoft copilot.

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

Yue Niu, Ramy E Ali, and Salman Avestimehr. 2022. 3legrace: Privacy-preserving dnn training over tees and gpus. *Proceedings on Privacy Enhancing Technologies*, 4:183–203.

OpenAI. Introducing chatgpt.

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

Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. 2023. From words to watts: Benchmarking the energy costs of large language model inference. *arXiv preprint arXiv:2310.03003*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*.

Zhiqing Sun, Yiming Yang, and Shinjae Yoo. 2021. Sparse attention with learning to hash. In *International Conference on Learning Representations*.

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

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

Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. 2020. Fast transformers with clustered attention. *Advances in Neural Information Processing Systems*, 33:21665–21674.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

A Lowrank Structure in Other Model

Figure 7 shows the low-rankness of BERT model on IMDB dataset. We can also observe that most sequences exhibists low-rank structures. In particular, long sequences are more low-rank, which is aligned with the observation in Sec 3.

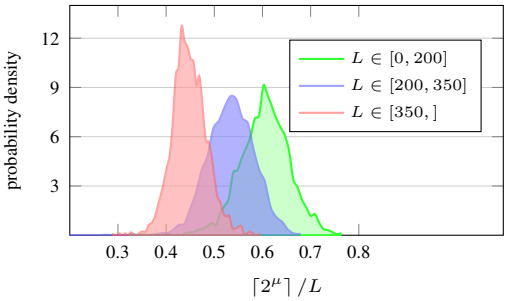


Figure 7: Distribution of low-rankness of BERT-base's embedding on IMDB dataset, measured by ratio $\lceil 2^{\mu} \rceil / L$.

B Alternating Optimization

Algorithm 1: Alternating Opt for SVD.

Data: $r, X, \{\boldsymbol{u}_i^0, \boldsymbol{v}_i^0\}_{i=1}^r$
Result: $\{\boldsymbol{u}_i, \boldsymbol{v}_i\}_{i=1}^r$
for i ***in*** $1, \cdots, r$ **do**
  for j ***in*** $1, \cdots, 2$ **do**
    /* Alternating optimization */
    $\boldsymbol{u}_i^j = \frac{X \cdot \boldsymbol{v}_i^{j-1}}{\|\boldsymbol{v}_i^{j-1}\|_F^2}$;
    $\boldsymbol{v}_i^j = \frac{X^T \cdot \boldsymbol{u}_i^j}{\|\boldsymbol{u}_i^j\|_F^2}$;
  end
  $\boldsymbol{u}_i, \boldsymbol{v}_i = \boldsymbol{u}_i^j, \boldsymbol{v}_i^{j^T}$;
  $X = X - \boldsymbol{u}_i^j \cdot \boldsymbol{v}_i^{j^T}$;
end

C Finetune Hyperparameters

For BERT-base and Llama2 models, we conduct a grid search on learning rate (1e-5, 2e-5, 5e-5, 1e-4, 2e-4, 5e-4), and weight decay (1e-3, 5e-3, 1e-2, 5e-2). Table 5 and 6 list the best hyperparameters found during fine-tuning.

max len	batch size	epochs	lr	wd
512	32	20	5e-5	1e-2

Table 5: Finetuning hyperparameters for BERT-base on SST-2, Squad, and IMDB.

max len	batch size	iters	lr	wd
2048	32	400	2e-4	1e-2

Table 6: Finetuning hyperparameters for Llama 2-7B/13B on MMLU and BoolQ.

D Prompt Format for MMLU and BoolQ

Table 7 and 8 list the prompt format for MMLU and BoolQ dataset.

```
The following are multiple choice questions (with answers).
One of the reasons that the government discourages
and regulates monopolies is that
A. producer surplus is lost and consumer surplus is gained.
B. monopoly prices ensure productive efficiency but cost
society allocative efficiency.
C. monopoly firms do not engage in significant research
and development.
D. consumer surplus is lost with higher prices and lower
levels of output.
Answer:
C
```

Table 7: MMLU prompt format

```
Below is an instruction that describes a task. Write a response
that appropriately completes the request.
### Instruction:
is harry potter and the escape from gringotts a roller coaster ride
### Input:
Harry Potter and the Escape from Gringotts is an indoor steel
roller coaster ***
### Answer:
Yes
```

Table 8: BoolQ prompt format