Proxyformer: Nyström-Based Linear Transformer with Trainable Proxy Tokens

Sangho Lee, Hayun Lee, Dongkun Shin*

Sungkyunkwan University
ilena7440@skku.edu, lhy920806@skku.edu, dongkun@skku.edu

Abstract

Transformer-based models have demonstrated remarkable performance in various domains, including natural language processing, image processing and generative modeling. The most significant contributor to the successful performance of Transformer models is the self-attention mechanism, which allows for a comprehensive understanding of the interactions between tokens in the input sequence. However, there is a well-known scalability issue, the quadratic dependency (*i.e.* $O(n^2)$) of self-attention operations on the input sequence length n, making the handling of lengthy sequences challenging. To address this limitation, there has been a surge of research on efficient transformers, aiming to alleviate the quadratic dependency on the input sequence length. Among these, the Nyströmformer, which utilizes the Nyström method to decompose the attention matrix, achieves superior performance in both accuracy and throughput. However, its landmark selection exhibits redundancy, and the model incurs computational overhead when calculating the pseudo-inverse matrix. We propose a novel Nyström method-based transformer, called Proxyformer. Unlike the traditional approach of selecting landmarks from input tokens, the Proxyformer utilizes trainable neural memory, called proxy tokens, for landmarks. By integrating contrastive learning, input injection, and a specialized dropout for the decomposed matrix, Proxyformer achieves top-tier performance for long sequence tasks in the Long Range Arena benchmark.

Introduction

Transformer-based models (Vaswani et al. 2017) such as BERT (Kenton and Toutanova 2019) and GPT-3 (Brown et al. 2020) have demonstrated state-of-the-art performance in various Natural Language Processing (NLP) tasks, including question answering (Rajpurkar et al. 2016), summarization (Miller 2019), and language modeling (Child et al. 2019). Recently, they have extended their influence to a wide range of applications, including image processing (Touvron et al. 2021) and generative modeling (Child et al. 2019). The most significant factor contributing to the successful performance of Transformer models is the *self-attention* mechanism. This key component enables transformer models to effectively understand interactions between tokens in the input sequence, model long-range dependencies, and capture contextual information from the entire sequence. However, a well-known scalability issue arises from the quadratic dependency (*i.e.,* $O(n^2)$) of self-attention operations on the input sequence length n, leading to slow and memory-intensive processing for long sequence inputs.

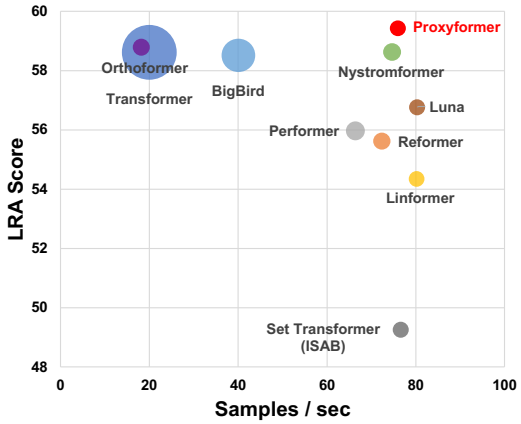

Figure 1: Performance on the Long Range Arena (LRA) benchmark (y axis), inference throughput (samples/sec) at a single GTX3090 GPU (x axis), and memory footprint (size of the circles) of various models on 4096 sequence length.

To tackle the scalability challenge, several *Efficient Transformers* (Tay et al. 2022) have been recently introduced, aiming to reduce the quadratic dependency to a sub-quadratic level. For example, BigBird (Zaheer et al. 2020) exploits the sparsity of attention to reduce the complexity of attention operation. Reformer (Kitaev, Kaiser, and Levskaya 2020) learns the sparse attention pattern in a data-driven fashion. Linformer (Wang et al. 2020) leverages the low-rankness of the attention map. Nyströmformer (Xiong et al. 2021) decomposes the softmax matrix of self-attention with the Nyström method (Wang and Zhang 2013). For an efficient approximation, the decomposed matrix uses landmarks sampled from the input sequence.

Although these techniques have all demonstrated successful optimization of self-attention operations, they have

*Corresponding author.

their respective limitations. Fig. 1 illustrates a comparative overview of the performance, inference throughput, and memory footprint of various existing efficient transformer models on the Long Range Arena (LRA) benchmark (Tay et al. 2021). BigBird achieves a better throughput than the standard transformer. However, its irregular sparse attention pattern leads to a limited throughput enhancement on real hardware platforms. Reformer, Linformer, and Performer have effectively optimized attention operations in terms of throughput and memory footprint. Nevertheless, they can be vulnerable to specific tasks and show lower accuracy compared to the standard transformer model.

The most successful technique to date for creating efficient transformers is Nyströmformer, which achieves superior results in both accuracy and inference throughput. We analyzed Nyströmformer in depth and found several limitations. Firstly, Nyströmformer selects landmarks by sampling (*i.e.,* average pooling) for the input sequence tokens without considering the redundancy among the selected landmarks. Secondly, the presence of a pseudo-inverse process during the approximation stage can lead to computation overhead.

We introduce an advanced variant of Nyströmformer, called **Proxyformer**. It introduces trainable special tokens, termed *proxy tokens*, to establish effective landmarks without sampling. These learned proxy tokens are used as landmarks to decompose attention operations with the Nyström method. To ensure minimal redundancy between these proxy tokens, we employ *contrastive loss* during their training. Additionally, we uses an *input injection module* at the first layer to adapt the learned tokens based on the information of input tokens. Each layer passes its own proxy token outputs to the following layer, ensuring every layer remains contextually aware of the input.

Similar approaches, such as Set Transformer (Lee et al. 2019) and Luna (Ma et al. 2021), have used learned tokens (inducing points) to decompose attention operations. However, these methods rely on two nested-attention blocks, requiring intricate layer designs. This deviates from the conventional structure of standard transformer models, hindering the adoption of pre-trained weights. Notably, Set Transformer lags in performance compared to Nyströmformer due to its inducing points being oblivious to input context, as shown in Fig. 1. While Luna seeks to rectify this by assimilating input context and transmitting the encoded data to subsequent layers, its performance remains similar to that of Nyströmformer. Conversely, Proxyformer retains the original transformer's structure by approximating the attention matrix using the Nyström method, which facilitates the use of pre-trained weights. Moreover, Proxyformer outperforms Nyströmformer in both accuracy and throughput.

Our main contributions are as follows:

- We integrate trainable proxy tokens with the Nyström approximation method to decompose the attention matrix, eliminating sampling overhead, achieving a linear dependency $O(n)$ on the input sequence length n, akin to Nyströmformer.
- Using contrastive loss and input injection, we generate input-aware proxy tokens that exhibit minimal redundancy.
- We present a dropout technique specifically designed for the Nyström-based decompsoed attention matrix.
- Our experiments demonstrate that Proxyformer achieves top-tier performance on the LRA benchmark and consistently performs well across all tasks.

Background and Motivation

Self-attention operation. The self-attention operation (Vaswani et al. 2017) stands as a pivotal element within transformer models. Fig. 2 illustrates a comparison of various attention techniques. In the conventional approach, shown in Fig. 2(a), each token attends to every other token, including itself. In the self-attention operation, the hidden states $X = (x_1, x_2, ..., x_n) \in \mathbf{R}^{n \times d}$ of an input sequence have n tokens in d dimensions. They are projected to query (Q), key (K), and value (V) by three weight matrices, $W_Q \in \mathbf{R}^{d \times d_q}$, $W_K \in \mathbf{R}^{d \times d_k}$, and $W_V \in \mathbf{R}^{d \times d_v}$ ($d_q = d_k$), respectively, as follows:

$$Q = XW_Q, K = XW_K, V = XW_V \qquad (1)$$
$$Q, K, V \in \mathrm{R}^{n \times d_{q,k,v}}$$

Then, the attention values between tokens are computed as:

$$Attention(Q, K, V) = \mathcal{S}\left(\frac{QK^T}{\sqrt{d_k}}\right)V = AV \qquad (2)$$

Here, $\mathcal{S}(\cdot)$ denotes the row-wise softmax normalization operation, and $A \in \mathrm{R}^{n \times n}$ is the attention matrix, which contains the attention weights between tokens. The self-attention operation represents the value of a token as a weighted sum of all token values, based on their attention weights. The computation in Eq. 2 exhibits a quadratic complexity of $O(n^2 d_k + n^2 d_v)$ for an input sequence length n. As the input sequence length grows, this computation turns into a significant computational and memory challenge for the transformer model, thereby restricting its ability to handle longer input sequences.

Efficient Transformers. Numerous studies have been conducted on efficient transformers with the goal of alleviating the quadratic dependency on input sequence length. The first approach is to exploit the **sparsity of attention** maps. It restricts attention computation to a specific sparse attention pattern. Local attention (Parmar et al. 2018) and Strided attention (Child et al. 2019) focus on the locality of attention maps and perform computations only within such fixed localized regions, as shown in Fig. 2(b).

To learn the sparse access pattern from the data set, Reformer (Kitaev, Kaiser, and Levskaya 2020) trains the query and key matrices to be clustered into buckets by locality sensitive hashing and considers the relationship between tokens only within the same bucket at attention operations, as depicted in Fig. 2(c). It reduces dot-product operations and achieves a complexity of $O(n\log n)$. There are also similar approaches using sorting or clustering-based methods (Tay et al. 2020; Vyas, Katharopoulos, and Fleuret 2020).

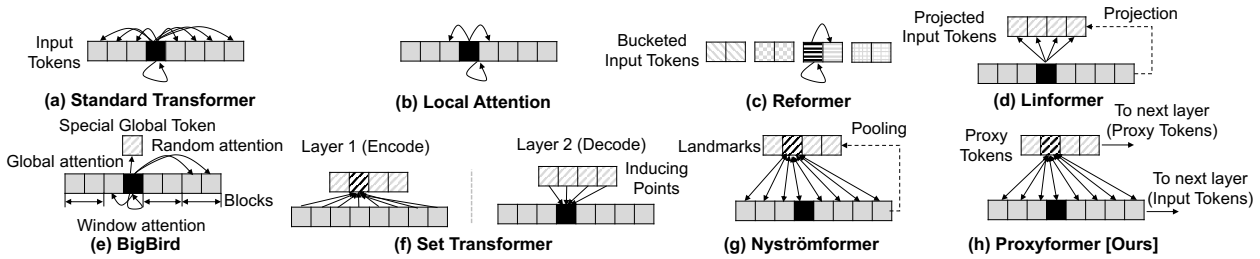

Figure 2: Comparison on different efficient transformers, including our Proxyformer.

The second approach is to leverage **low-rank approximation** and **kernelization**. Linformer (Wang et al. 2020) projects the key and value matrices to a reduced dimension to exploit the low-rankness of self-attention, as illustrated in Fig. 2(d), thereby reducing the complexity to $O(n)$. Performer (Choromanski et al. 2021) achieves linear complexity $O(n)$ by approximating the softmax attention kernel using orthogonal random features.

Lastly, there are **neural memory** approaches. The neural memory has additional trainable special tokens to comprehend the entire sequence with a reduced computational complexity. The neural memory was originally used by BERT (Kenton and Toutanova 2019) and Vision Transformer (Dosovitskiy et al. 2021) as a form of **[CLS]** tokens to aggregate information from the entire context. To address the limitation of local attention at fixed sparse pattern techniques, Longformer (Beltagy, Peters, and Cohan 2020) and BigBird (Zaheer et al. 2020) incorporated additional 'global attention' tokens as a neural memory to enable effective computation on global contextual information. Particularly, BigBird combined window-based local attention, global attention, and random attention, as shown in Fig. 2(e). Set Transformer (Lee et al. 2019), illustrated in Fig. 2(f), also uses the neural memory technique. It replaces self-attention with two cross-attentions between the learned 'inducing points' and the input sequence. The input sequence is encoded and subsequently decoded, effectively optimizing self-attention computations.

Nyströmformer (Xiong et al. 2021) uses a **downsampling** approach rather than training a neural memory. It derives landmarks from input tokens via pooling, which are then employed to reconstruct the softmax matrix in an approximated self-attention mechanism. Given that the count of landmarks is considerably less than the sequence length, a linear complexity is achieved. Nonetheless, the method employed for landmark sampling is susceptible to redundancy. Furthermore, Nyströmformer incurs a significant computational overhead when calculating the pseudo-inverse of the attention matrix between the landmarks.

As depicted in Fig. 2(h), our Proxyformer integrates the Nyström method with neural memory (proxy tokens), rather than relying on landmark sampling. While Nyströmformer discards intermediate results generated with landmarks after the attention matrix approximation, Proxyformer conveys the data from the proxy tokens to the next layer. As a result, only the initial layer utilizes the pre-learned proxy tokens, with succeeding layers inheriting their landmarks from the preceding layer.

Proxy Token-Based Linear Transformer

In this section, we begin with a concise overview of Nyström-based attention, which employs the Nyström method for self-attention approximation. Subsequently, we delve into Proxyformer and elaborate on its contrastive loss, input injection, and attention dropout strategies.

Nyström-Based Self-Attention

The Nyström method is used by Nyströmformer (Xiong et al. 2021) and Orthoformer (Patrick et al. 2021) for approximating the full attention matrix A. The key idea of this method is to sample landmark vectors $\tilde{Q}$ and $\tilde{K}$ from from the matrices Q and K, which originate from the input sequence. The full attention matrix is decomposed with these landmarks to approximate it.

Given the query matrix $Q = [q_1, \ldots, q_n] \in \mathbf{R}^{n \times d_q}$ and the key matrix $K = [k_1, \ldots, k_n] \in \mathbf{R}^{n \times d_k}$, the landmark matrices, $\tilde{Q}$ and $\tilde{K}$, are derived, through a sampling operation on Q and K such as strided-pooling.

$$\tilde{Q} = [\tilde{q_1}, \cdots, \tilde{q_m}] \in \mathbf{R}^{m \times d_q} \tag{3}$$

$$\tilde{K} = [\tilde{k_1}, \cdots, \tilde{k_m}] \in \mathbf{R}^{m \times d_k} \tag{4}$$

$\tilde{q_j} \in \mathbf{R}^{1 \times d_q}$ and $\tilde{k_j} \in \mathbf{R}^{1 \times d_k}$ represent the determined landmarks. Utilizing the Nyström method, the approximated attention matrix $\tilde{A} \in R^{n \times n}$ can be represented as:

$$\tilde{A} = \left[\mathcal{S}\left(\frac{Q\tilde{K}^T}{\sqrt{d_q}}\right)\right]_{n \times m} [A_s^+]_{m \times m} \left[\mathcal{S}\left(\frac{\tilde{Q}K^T}{\sqrt{d_q}}\right)\right]_{m \times n} \tag{5}$$

Here, A_s^+ is Moore-Penrose inverse of $A_s = \mathcal{S}\left(\frac{\tilde{Q}\tilde{K}^T}{\sqrt{d_q}}\right)$.

The self-attention operation can be efficiently approximated by computing the product of the attention matrix $\tilde{A}$ and the value matrix V.

$$AV \approx \tilde{A}V = \mathcal{S}\left(\frac{Q\tilde{K}^T}{\sqrt{d_q}}\right) A_s^+ \mathcal{S}\left(\frac{\tilde{Q}K^T}{\sqrt{d_q}}\right) V \tag{6}$$

The Nyström-based approximated self-attention operation has a complexity of $O(nm)$, where m is the number of landmarks. When $m \ll n$, it exhibits linear dependency with $O(n)$ (Xiong et al. 2021).

Algorithm 1 Nyströmformer attention

procedure NYSTRÖMFORMER($\mathbf{Q}, \mathbf{K}, \mathbf{V}$)
  $\mathbf{L_q}, \mathbf{L_k} \leftarrow \text{SegmentMeans}(\mathbf{Q}, \mathbf{K}, m)$
  $\mathcal{K}_1 = \mathcal{S}(\mathbf{Q}\mathbf{L_k}^{\mathrm{T}}/\sqrt{d_k})$
  $\mathcal{K}_2 = \mathcal{S}(\mathbf{L_q}\mathbf{L_k}^{\mathrm{T}}/\sqrt{d_k})$
  $\mathcal{K}_2^{+} = \text{Iterative\_Inverse}(\mathcal{K}_2, N_{iter})$
  $\mathcal{K}_3 = \mathcal{S}(\mathbf{L_q}\mathbf{K}^{\mathrm{T}}/\sqrt{d_k})$
  $\mathbf{Y} = \mathcal{K}_1(\mathcal{K}_2^{+}(\mathcal{K}_3\mathbf{V}))$
end procedure

Algorithm 2 Orthoformer attention

procedure ORTHOFORMER($\mathbf{Q}, \mathbf{K}, \mathbf{V}$)
  $\mathbf{L} \leftarrow \text{MostOrthogonalSubset}(\mathbf{Q}, \mathbf{K}, m)$
  $\mathcal{K}_1 = \mathcal{S}(\mathbf{Q}\mathbf{L}^{\mathrm{T}}/\sqrt{d_k})$
  $\mathcal{K}_2 = \mathcal{S}(\mathbf{L}\mathbf{K}^{\mathrm{T}}/\sqrt{d_k})$
  $\mathbf{Y} = \mathcal{K}_1(\mathcal{K}_2\mathbf{V})$
end procedure

Algorithm 3 Proxyformer attention

procedure PROXYFORMER($\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{L}$)
  $\mathcal{K}_1 = \mathcal{S}(\mathbf{Q}\mathbf{L}^{\mathrm{T}}/\sqrt{d_k})$
  $\mathcal{K}_2 = \mathcal{S}(\mathbf{L}\mathbf{K}^{\mathrm{T}}/\sqrt{d_k})$
  $\mathbf{Y_p} = \mathcal{K}_2\mathbf{V}$
  $\mathbf{Y_x} = \mathcal{K}_1\mathbf{Y_p}$
  $\mathbf{Y} = [\mathbf{Y_p}; \mathbf{Y_x}]$
end procedure

Figure 3: Comparision of Nyström-based self-attention variants

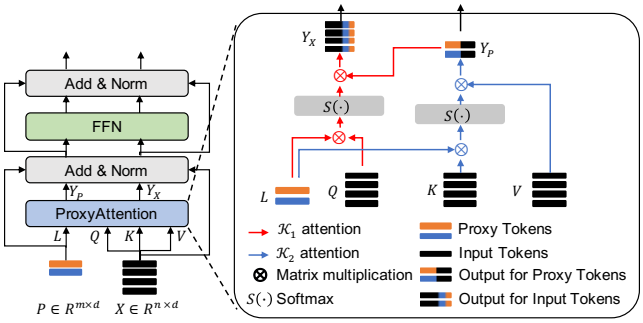

Figure 4: Transformer encoder of Proxyformer.

Algorithm 1 and Algorithm 2 in Fig. 3 represent the attention algorithms of Nyströmformer and Orthoformer, respectively. These algorithms differ in two main operational aspects. Firstly, while Nyströmformer uses strided-based pooling (SegmentMeans) from queries and keys to determine m landmarks, Orthoformer builds orthogonal landmarks to minimize the redundancy among landmarks. It is done by incremental landmark selection. At each step, an orthogonal landmark is selected based on its cosine similarity with landmarks chosen in prior steps. Secondly, while Nyströmformer employs a pseudo-inverse of A_s ($\mathcal{K}_2^{+}$ in Algorithm 1), which is determined through an iterative approximation technique, instead of calculating $\mathcal{A}_s^{+}$, Orthoformer does not require the calculation of $\mathcal{A}_s^{+}$ because its landmarks are orthogonal.

Orthoformer enhances self-attention performance by reducing redundancy among landmarks and eliminating the need for inverse calculations. However, its orthogonal landmark searching method incurs significant runtime overhead. Consequently, as illustrated in Fig. 1, Orthoformer's throughput is not better than that of the standard transformer. In contrast, Proxyformer employs trainable proxy tokens instead of input-based landmarks. This allows for the orthogonality of proxy tokens to be established during model training, facilitating a more efficient and effective construction of the landmarks.

Proxy self-attention. Fig. 4 illustrates the structure of Proxyformer encoder, which consists of alternating layers of Proxy attention blocks and FFN blocks, following the typical encoder structure of Transformer model (Vaswani et al. 2017). In addition to the input sequence X, Proxyformer employs m trainable proxy tokens $P = (p_0, \ldots, p_m) \in \mathrm{R}^{m \times d}$. The proxy tokens are fed into the proxy attention block along with the input tokens X. The proxy self-attention performs the Nyström attention using landmarks constructed by trained proxy tokens. Algorithm 3 in Fig. 3 shows the algorithm of proxy attention. While the input token sets X is projected to Q, K, and V, the m proxy token sets P is projected to landmarks $L \in \mathrm{R}^{m \times d_l}$ through $W_L \in \mathrm{R}^{d \times d_l}$ ($d_l = d_k = d_q$), *i.e.*, $L = PW_L$.

For the Nyström-based attention operation, Proxyformer also uses the same approximation operation in Eq. (6). However, the landmark L substitutes $\tilde{Q}$ and $\tilde{K}$. To approximate the attention in a decomposed form, we computes $\mathcal{K}_1 \in \mathrm{R}^{n \times m}$ through operations between Q and L, and $\mathcal{K}_2 \in \mathrm{R}^{m \times n}$ through operations between K and L. The operation between $\mathcal{K}_2$ and V compute the compressed form of V represented as $\mathrm{Y_P} \in \mathrm{R}^{m \times d_v}$. Finally, the operation between $\mathcal{K}_1$ and $\mathrm{Y_P}$ yields the output $\mathrm{Y_X} \in R^{n \times d_v}$ for the input sequence. While Nyströmformer discards the intermediate result on landmarks ($\mathcal{K}_3 V$ in Algorithm 1) after the attention operation, Proxyformer passes it ($\mathrm{Y_p}$ in Algorithm 3) to the next layer, and the subsequent layer uses it as the proxy token set for the layer.

Contrastive landmarks. To approximate the attention matrix using a minimal set of landmarks in the Nystörm-based attentions, it's crucial to ensure minimal redundancy among these landmarks. Orthoformer achieves this by incrementally selecting the most orthogonal subset from the query and key set. Starting with a randomly chosen landmark, each subsequent landmark is picked to be as orthogonal as possible to the previously selected ones. This process, however, demands repeated cosine similarity computations during the runtime. Moreover, the incremental greedy search could not find the optimal landmarks. On the other hand, Proxyformer uses a contrastive loss to train proxy tokens. This approach facilitates the creation of landmarks with minimal redundancy, without introducing additional runtime overhead.

Contrastive learning (Hadsell, Chopra, and LeCun 2006) can learn representations by contrasting positive pairs against negative pairs. We adopt cosine similarity as the similarity function between two landmarks and utilize the contrastive loss, denoted as L^c, to guide the learning towards separating the directions of each two landmarks, as proposed in SupContrast (Khosla et al. 2020). The model is trained

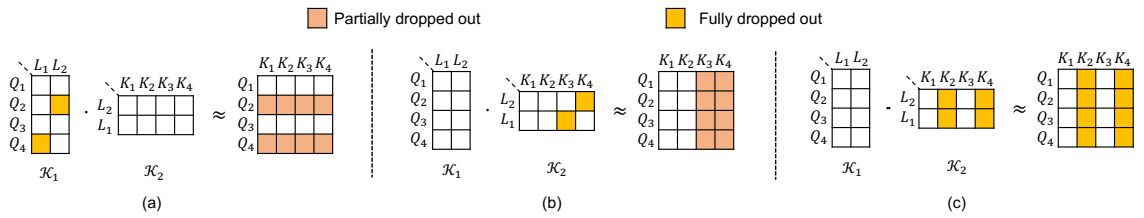

Figure 5: Dropout on decomposed attention matrix. (a) represents normal dropout on $\mathcal{K}_1$ matrix. (b) represents normal dropout on $\mathcal{K}_2$ matrix. (c) represents column dropout on $\mathcal{K}_2$ matrix (proposed decomposed dropout).

using both the cross-entropy loss $\mathcal{L}_{CE}$ for the task and the contrastive loss $\mathcal{L}^c$.

$$z_i = L_i / ||L_i||_2, \quad i = 1, 2, \ldots, m \tag{7}$$

$$\mathcal{L}^c = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}_i^c = -\sum_{i=1}^{m}\log\frac{\exp(z_i \cdot z_i/\tau)}{\sum_{j=1}^{m}\exp(z_i \cdot z_j/\tau)} \tag{8}$$

$$\mathcal{L}_{tot} = \mathcal{L}_{CE} + \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_t^c, \quad t = 1, \cdots, T \tag{9}$$

Here, $z_i \in R^d$ represents the i-th normalized landmarks indicating the direction. t denotes the layer index, and $\tau \in R^+$ representing a scalar temperature parameter.

Contrastive learning has an additional advantage. We can eliminate the iterative computation for the pseudo inverse of A_s^+ of Nyströmformer. Because the landmarks are trained to reduce the cosine similarity between each other, the matrix $A_s \in \mathrm{R}^{m\times m}$ computes closely to the identity matrix $\mathrm{I} \in \mathrm{R}^{m\times m}$. This enables us to circumvent operations associated with A_s^+.

$$A_s = \mathcal{S}(\mathbf{L}\mathbf{L}^{\mathbf{T}}/\sqrt{d_q}) \approx \mathbf{I}, \quad A_s^+ \approx \mathbf{I} \tag{10}$$

Injection module. In Proxyformer, each layer generates the proxy tokens for the next layer, accumulating information about the input sequence. This enables the construction of effective landmarks that take the input context into account. However, in the first encoder block, attention is approximated solely based on trained tokens, which fails to incorporate the information from the input context, leading to a performance degradation. To address this issue, we use an injection module to the embedding layer, aiming to provide the input context information to the proxy tokens.

Patch Merger (Renggli et al. 2022), a computation-efficient vision transformers, compresses n input patches X into m sequences before passing them to the next encoder. This is achieved through cross-attention operations with a trainable weight matrix $W \in \mathrm{R}^{m\times d}$. We leverage the idea of Patch Merger to design the injection module that effectively injects input sequence information into the proxy tokens. The injection module generates m compressed tokens from the input sequence X through cross-attention with the learned proxy tokens P, and subsequently injections them into the proxy tokens via a residual connection:

$$P = P + \mathcal{S}\left(PX^T/\sqrt{d}\right)X \tag{11}$$

The injection module has a minimal impact on the overall computational cost while successfully injecting information about the input sequence to the proxy tokens.

Dropout for decomposed matrix. Dropout (Srivastava et al. 2014) is a regularization technique used to prevent the model from overfitting on specific features. In Transformer models, dropout is applied to the weights computed by the Softmax function in the attention mechanism. It randomly sets some weights to zero, preventing excessive reliance on specific token relationships and enabling the model to extract information from various perspectives (Zehui et al. 2019). There are two decomposed weight matrices, $\mathcal{K}_1$ and $\mathcal{K}_2$, in Proxyformer. Implementing a conventional dropout method to either, as depicted in Fig. 5(a) and (b), results in merely a partial dropout of some weight values (*i.e.,* they are not zeroed.) in the reconstructed attention matrix. This does not fully eliminate the inter-token relationships. In addition, if the dropout is applied to $\mathcal{K}_1$, all keys within specific rows of the reconstructed attention matrix are affected, making distorted relationships. With this in mind, we propose two effective dropout approaches for decomposed matrices:

- Dropout is applied on $\mathcal{K}_2$ in Algorithm 3 so that not all keys related to a particular query are affected.
- Dropout is employed to the full column of chosen keys to completely remove certain weights within the reconstructed matrix.

By applying a column-wise dropout only on the $\mathcal{K}_2$ matrix, as shown in Fig. 5(c), we successfully achieve the regularization effect on the decomposed attention matrix.

Complexity analysis of Proxyformer. In Proxyformer, for each layer, there is a linear projection for proxy tokens and an additional operation by the FFN. The complexity of the linear projection depends on the number of landmarks, denoted as m, which results in a linear complexity of $O(m)$. The proxy attention operation of Proxyformer involves the computation of $\mathcal{S}(QL^T/\sqrt{d_k})(\mathcal{S}(LK^T/\sqrt{d_k})V)$, and the complexity of this operation is $O(nmd_v + mnd_v)$.

Regarding memory footprint, Proxyformer requires additional memory for landmarks, whose size is proportional to md_l. Additionally, it needs to store three matrices for Nyström approximation, which require memory space proportional to $nm+mn+nd_v$. Therefore, the overall memory complexity is $O(md_l + mn + nm + nd_v)$.

Given that the number of landmarks m is much smaller than n, the computational and memory complexities of Proxyformer are both $O(n)$, scaling linearly w.r.t. the input sequence length n.

Model	ListOps (n=2K)	Text (n=4K)	Retrieval (n=4K)	Image (n=1K)	Pathfinder (n=1K)	AVG.
Standard Transformer (Vaswani et al. 2017)*	37.10	65.02	79.35	38.20	**74.16**	58.77
Standard Transformer(re-impl)	37.09	64.84	80.01	38.57	72.61	58.62
Reformer-2* (Kitaev, Kaiser, and Levskaya 2020)	19.05	64.88	78.64	**43.29**	69.36	55.04
Linformer-256* (Wang et al. 2020)	37.25	55.91	79.37	37.84	67.60	55.59
Performer-256* (Choromanski et al. 2021)	18.80	63.81	78.62	37.07	69.87	53.63
Nyströmformer-256* (Xiong et al. 2021)	37.15	65.52	79.56	41.58	70.94	**58.95**
Linformer-128 (Wang et al. 2020)	**37.92**	54.59	77.69	34.91	66.63	54.35
Performer-128 (Choromanski et al. 2021)	28.38	64.67	79.85	38.19	68.77	55.97
Set Trasnformer-ISAB-128 (Lee et al. 2019)	18.52	63.63	77.80	36.39	49.97	49.26
Bigbird (Zaheer et al. 2020)	37.23	63.94	**80.44**	39.46	71.53	58.52
Nyströmformer-128	37.18	**65.73**	**80.44**	38.64	71.18	58.63
Orthoformer-128 (Patrick et al. 2021)	37.16	65.12	79.53	39.69	72.48	58.80
Luna-128 (Ma et al. 2021)	37.29	64.47	78.12	34.84	69.11	56.77
Proxyformer-128 (ours)	37.22	65.49	80.23	40.14	74.07	**59.43**

Table 1: Classification accuracy results on Long Range Arena (LRA) benchmark. The symbol of $*$ denotes 256 projection dimension results reported at Nyströmformer (Xiong et al. 2021). The best result for each benchmark task is bolded.

Experiments

To compare the performance of Proxyformer with those of other efficient transformer models, we carried out experiments using the Long Range Arena (LRA) benchmark (Tay et al. 2021). The LRA benchmark is commonly utilized to assess the capability of transformer models in capturing long-range dependencies. We present the average accuracy, memory usage, and computational efficiency of various transformers in scenarios involving long sequences.

Implementation Details

We used Nyströmformer's LRA PyTorch implementation, which employs a two-layer transformer model with 64 embedding dimensions, 128 feed-forward dimensions, and two attention heads. To ensure similar computational complexity across all variants, we set the projection dimension (*e.g.,* # of proxy tokens and # of landmarks) to 128 for all projection-based variants. For Reformer and Bigbird, we used 2-hashing functions and a block size of 64, respectively. The temperature parameter for contrastive loss and dropout probability were set to 0.07 and 0.1, respectively. Furthermore, to observe only the capability of the self-attention operation, we removed the convolution module from the Nyströmformer implementation, which has been employed to enhance the performance on vision task.

For experiments focusing on inference-time efficiency, we used a more extensive transformer model featuring six encoder layers, following the structure of the standard transformer. We recorded the memory usage per sequence and throughput on a single NVIDIA GeForce RTX 3090 GPU.

Results: Accuracy and Efficiency

Table 1 illustrates the accuracy of various transformers, and Table 2 presents the efficiency results. The reported accuracy is the average value of five random seed experiments. Proxyformer exhibits superior accuracy compared to other self-attention operations, demonstrating robustness across all LRA tasks. This is attributed to the Proxyformer's ability to encapsulate the context within the proxy tokens at each layer. In particular, it shows 0.8% higher accuracy compared to the Nyströmformer-128. (For comparison, Nyströmformer-256 is a convolution-enabled version.)

Sequence Length	2048		4096	
Model	MEM↓	TP↑	MEM↓	TP↑
Standard	1(286.8MB)	1(65)	1(1091.7MB)	1(20)
Set Transformer	**0.16**	2.32	0.09	3.83
Reformer	0.19	2.21	0.10	3.62
Linformer	0.16	2.46	**0.08**	4.01
Performer	0.23	2.04	0.12	3.32
Bigbird	0.69	1.24	0.37	2.00
Nyströmformer	0.20	2.19	0.11	3.74
Orthoformer	0.17	0.56	0.10	0.91
Luna	**0.16**	**2.56**	**0.08**	**4.22**
Proxyformer	**0.17**	**2.27**	**0.08**	**3.80**

Table 2: Average memory footprint per one input sequence (MEM) and throughput (TP) results for 2K and 4K input sequence lengths. The throughput means the number of inputs per seconds. These values are normalized to those of the standard transformer. The absolute result values of the standard transformer are provided in parentheses.

Proxyformer shows a substantial decrease in memory footprint, 83% and 92% reductions compared to the standard transformer for 2048 and 4096 sequence lengths, respectively. The throughput increases by $2.27\times$ and $3.8\times$ for the mentioned sequence lengths. Notably, as the sequence length increases, its efficiency becomes more significant owing to its linear complexity w.r.t. the sequence length n.

Although Orthoformer achieves higher accuracy (+0.17%) than Nyströmformer due to the orthogonal landmark sampling, this comes with a notable sampling

Model			ListOps (n=2K)	Text (n=4K)	Retrieval (n=4K)	Image (n=1K)	Pathfinder (n=1K)	AVG.
Dropout	Contrastive	Injection	Train/Test	Train/Test	Train/Test	Train/Test	Train/Test	Train/Test
-	-	-	39.2/36.9	84.6/65.2	86.6/80.1	66.7/39.0	80.9/72.5	71.6/58.7
-	✓	-	36.1/37.0	82.3/65.4	86.7/80.4	63.4/38.7	79.4/73.0	69.6/58.9
Normal on $\mathcal{K}_1$	-	-	36.0/36.9	84.3/65.1	86.2/80.2	66.6/38.9	81.0/72.8	70.8/58.8
Normal on $\mathcal{K}_2$	-	-	39.1/37.0	84.8/64.8	86.4/80.3	66.9/39.5	79.8/70.2	71.4/58.4
Decomposed	-	-	37.5/37.0	81.6/65.2	85.2/80.0	63.7/40.1	80.0/73.3	69.5/59.2
Decomposed	✓	-	34.3/37.0	78.9/65.5	85.0/80.1	61.1/40.2	77.7/73.6	67.4/59.3
Decomposed	✓	✓	37.1/37.2	80.0/65.5	85.6/80.2	61.9/40.1	78.5/74.1	**68.6/59.4**

Table 3: Ablation study using the Long Range Arena (LRA) benchmark. We evaluate the performance of the proposed decomposed dropout, contrastive loss, and injection module. The results showcase the train/test classification accuracy for each task. The train accuracy represents the average accuracy over the last 10 batches.

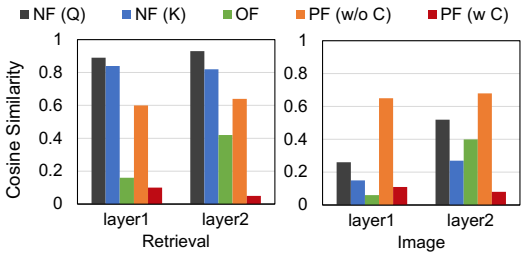

Figure 6: Average of cosine similarity between landmarks on Retrieval and Image test dataset. NF: Nyströmformer, OF: Orthoformer, PF: Proxyformer, C: contrastive loss

overhead, as observed in Table 2. However, Proxyformer minimizes redundancy via contrastive loss, ensuring efficient computations without runtime overhead. It also attains lower memory footprint and higher throughput than Nyströmformer by eliminating pseudo-inverse operations.

We analyzed the cosine similarity among landmarks for the Nyströmformer, Orthoformer, and Proxyformer, as depicted in Fig. 6. The Nyströmformer displays significant redundancy, particularly evident in the Retrieval tasks. While Orthoformer mitigates this redundancy using its landmark selection algorithm, the results remain less than ideal. Proxyformer, on the other hand, significantly reduces redundancy via contrastive learning. Without the application of contrastive learning in the Proxyformer, the proxy tokens tend to exhibit high redundancy. With contrastive learning, Proxyformer consistently delivers superior results across all tasks.

Despite Linformer (Wang et al. 2020) and Luna (Ma et al. 2021) being more efficient than Proxyformer, they show inferior accuracies compared to Proxyformer (-5.08% and -2.66%). Proxyformer facilitates efficient self-attention operations, successfully balancing efficiency and accuracy.

Ablation Study

Table 3 demonstrates the average accuracy of Proxyformer under various settings. Without dropout or contrastive loss, proxy tokens tend to focus on specific input tokens, resulting in a test accuracy that is notably lower than its training counterpart. Introducing the contrastive loss causes a dip in training accuracy as it inhibits the model from becoming too aligned with a particular task. Yet, this results in a 0.2% increase in test accuracy. This underscores the idea that minimizing redundancy in landmarks serves as an effective regularization technique, ultimately improving the model's overall performance.

When applying standard dropout (Srivastava et al. 2014) to the $\mathcal{K}_1$ matrix, the decomposed matrix was not effectively regularized, making the benefits of regularization barely noticeable. Applying standard dropout to the $\mathcal{K}_2$ matrix did present some regularization effects. Yet, this led to a substantial accuracy decline for the Pathfinder task, causing an overall reduction in average accuracy. Conversely, the decomposed dropout resulted in roughly a 0.5% accuracy boost, showcasing the effect of dropout regularization.

Combining both contrastive loss and decomposed dropout improved overfitting across most tasks, leading to about a 0.6% increase in test accuracy. Furthermore, integrating the injection module led to a rise in both training and testing accuracy (+1.2%/+0.1%), underscoring its efficacy in crafting input-sensitive landmarks.

Conclusion

We presented Proxyformer, a Nyström-based model that uses trainable proxy tokens as landmarks, facilitating efficient attention with linear complexity. Our findings indicate that landmarks for Nyström-based decomposition can be effectively trained to minimize redundancy and maintain sensitivity to input context. This is achieved through the incorporation of contrastive loss, injection module, and tailored dropout. Proxyformer not only showcased leading performance on LRA tasks but also optimized computation and memory usage.

In future research, we plan to extend our validation of Proxyformer's effectiveness by integrating it with various transformer models like BERT (Kenton and Toutanova 2019) and ViT (Dosovitskiy et al. 2021). Furthermore, we intend to delve into challenges that may arise when applying proxy self-attention during the fine-tuning phase of pretrained models, especially focusing on the initialization of proxy tokens.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00421, AI Graduate School Support Program(Sungkyunkwan University)) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.IITP-2017-0-00914, Software Framework for Intelligent IoT Devices)

References

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Child, R.; Gray, S.; Radford, A.; and Sutskever, I. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Choromanski, K. M.; Likhosherstov, V.; Dohan, D.; Song, X.; Gane, A.; Sarlos, T.; Hawkins, P.; Davis, J. Q.; Mohiuddin, A.; Kaiser, L.; Belanger, D. B.; Colwell, L. J.; and Weller, A. 2021. Rethinking Attention with Performers. In *International Conference on Learning Representations*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Hadsell, R.; Chopra, S.; and LeCun, Y. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, 1735–1742. IEEE.

Kenton, J. D. M.-W. C.; and Toutanova, L. K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, 2.

Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; and Krishnan, D. 2020. Supervised contrastive learning. *Advances in neural information processing systems*, 33: 18661–18673.

Kitaev, N.; Kaiser, L.; and Levskaya, A. 2020. Reformer: The Efficient Transformer. In *International Conference on Learning Representations*.

Lee, J.; Lee, Y.; Kim, J.; Kosiorek, A.; Choi, S.; and Teh, Y. W. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, 3744–3753. PMLR.

Ma, X.; Kong, X.; Wang, S.; Zhou, C.; May, J.; Ma, H.; and Zettlemoyer, L. 2021. Luna: Linear unified nested attention. *Advances in Neural Information Processing Systems*, 34: 2441–2453.

Miller, D. 2019. Leveraging BERT for extractive text summarization on lectures. *arXiv preprint arXiv:1906.04165*.

Parmar, N.; Vaswani, A.; Uszkoreit, J.; Kaiser, L.; Shazeer, N.; Ku, A.; and Tran, D. 2018. Image transformer. In *International conference on machine learning*, 4055–4064. PMLR.

Patrick, M.; Campbell, D.; Asano, Y.; Misra, I.; Metze, F.; Feichtenhofer, C.; Vedaldi, A.; and Henriques, J. F. 2021. Keeping your eye on the ball: Trajectory attention in video transformers. *Advances in neural information processing systems*, 34: 12493–12506.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2383–2392. Austin, Texas: Association for Computational Linguistics.

Renggli, C.; Pinto, A. S.; Houlsby, N.; Mustafa, B.; Puigcerver, J.; and Riquelme, C. 2022. Learning to merge tokens in vision transformers. *arXiv preprint arXiv:2202.12015*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958.

Tay, Y.; Bahri, D.; Yang, L.; Metzler, D.; and Juan, D.-C. 2020. Sparse sinkhorn attention. In *International Conference on Machine Learning*, 9438–9447. PMLR.

Tay, Y.; Dehghani, M.; Abnar, S.; Shen, Y.; Bahri, D.; Pham, P.; Rao, J.; Yang, L.; Ruder, S.; and Metzler, D. 2021. Long Range Arena : A Benchmark for Efficient Transformers. In *International Conference on Learning Representations*.

Tay, Y.; Dehghani, M.; Bahri, D.; and Metzler, D. 2022. Efficient Transformers: A Survey. arXiv:2009.06732.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2021. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, 10347–10357. PMLR.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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

Wang, S.; Li, B. Z.; Khabsa, M.; Fang, H.; and Ma, H. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Wang, S.; and Zhang, Z. 2013. Improving CUR matrix decomposition and the Nyström approximation via adaptive sampling. *The Journal of Machine Learning Research*, 14(1): 2729–2769.

Xiong, Y.; Zeng, Z.; Chakraborty, R.; Tan, M.; Fung, G.; Li, Y.; and Singh, V. 2021. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 14138–14148.

Zaheer, M.; Guruganesh, G.; Dubey, K. A.; Ainslie, J.; Alberti, C.; Ontanon, S.; Pham, P.; Ravula, A.; Wang, Q.; Yang, L.; et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33: 17283–17297.

Zehui, L.; Liu, P.; Huang, L.; Chen, J.; Qiu, X.; and Huang, X. 2019. Dropattention: A regularization method for fully-connected self-attention networks. *arXiv preprint arXiv:1907.11065*.