Long-Short Transformer: Efficient Transformers for Language and Vision

Chen Zhu[‡1*], Wei Ping[†2], Chaowei Xiao[†2,3], Mohammad Shoeybi[2], Tom Goldstein[1], Anima Anandkumar[2,4], and Bryan Catanzaro[2]

[1]University of Maryland, College Park
[2] NVIDIA [3]Arizona State University [4]California Institute of Technology
[‡]chenzhu@cs.umd.edu, [†]{wping, chaoweix}@nvidia.com

Abstract

Transformers have achieved success in both language and vision domains. However, it is prohibitively expensive to scale them to long sequences such as long documents or high-resolution images, because self-attention mechanism has quadratic time and memory complexities with respect to the input sequence length. In this paper, we propose Long-Short Transformer (Transformer-LS), an efficient self-attention mechanism for modeling long sequences with linear complexity for both language and vision tasks. It aggregates a novel long-range attention with dynamic projection to model distant correlations and a short-term attention to capture fine-grained local correlations. We propose a dual normalization strategy to account for the scale mismatch between the two attention mechanisms. Transformer-LS can be applied to both autoregressive and bidirectional models without additional complexity. Our method outperforms the state-of-the-art models on multiple tasks in language and vision domains, including the Long Range Arena benchmark, autoregressive language modeling, and ImageNet classification. For instance, Transformer-LS achieves 0.97 test BPC on enwik8 using half the number of parameters than previous method, while being faster and is able to handle $3\times$ as long sequences compared to its full-attention version on the same hardware. On ImageNet, it can obtain the state-of-the-art results (e.g., a moderate size of 55.8M model solely trained on 224×224 ImageNet-1K can obtain Top-1 accuracy 84.1%), while being more scalable on high-resolution images. The source code and models are released at https://github.com/NVIDIA/transformer-ls.

1 Introduction

Transformer-based models [1] have achieved great success in the domains of natural language processing (NLP) [2, 3] and computer vision [4–6]. These models benefit from the self-attention module, which can capture both adjacent and long-range correlations between tokens while efficiently scaling on modern hardware. However, the time and memory consumed by self-attention scale quadratically with the input length, making it very expensive to process long sequences. Many language and vision tasks benefit from modeling long sequences. In NLP, document-level tasks require processing long articles [e.g., 7, 8], and the performance of language models often increases with sequence length [e.g., 9, 10]. In computer vision, many tasks involve high-resolution images, which are converted to long sequences of image patches before being processed with Transformer models [4, 6, 11]. As a result, it is crucial to design an efficient attention mechanism for long sequence modeling that generalizes well across different domains.

*Work done during an internship at NVIDIA.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

Various methods have been proposed to reduce the quadratic cost of full attention. However, an efficient attention mechanism that generalizes well in both language and vision domains is less explored. One family of methods is to sparsify the attention matrix with predefined patterns such as sliding windows [e.g., 12–15] and random sparse patterns [16]. These methods leverage strong inductive biases to improve both computational and model performance, but they limit the capacity of a self-attention layer because each specific token can only attend to a subset of tokens. Another family of methods leverages low-rank projections to form a low resolution representation of the input sequence, but the successful application of these methods has been limited to certain NLP tasks [e.g., 17–19]. Unlike sparse attention, this family of methods allows each token to attend to the entire input sequence. However, due to the loss of high-fidelity token-wise information, their performance sometimes is not as good as full attention or sparse attention on tasks that require fine-grained local information, including standard benchmarks in language [20] and vision [21].

Despite the rapid progress in efficient Transformers, some proposed architectures can only be applied to bidirectional models [e.g., 15, 16, 18]. Transformer-based autoregressive models have achieved great successes in language modeling [22], image synthesis [23], and text-to-image synthesis [24], which also involve long texts or high-resolution images. It is desirable to design an efficient transformer that can be applied to both autoregressive and bidirectional models.

In this work, we unify a local window attention and a novel long-range attention into a single efficient attention mechanism. We show that these two kinds of attention have complementary effects that together yield the state-of-the-art results on a range of tasks in language and vision, for both autoregressive and bidirectional models. Specifically, we make the following contributions:

- We propose Long-Short Transformer (Transformer-LS), an efficient Transformer that integrates a dynamic projection based attention to model long-range correlations, and a local window attention to capture fine-grained correlations. Transformer-LS can be applied to both autoregressive and bidirectional models with linear time and memory complexity.
- We compute a dynamic low-rank projection, which depends on the content of the input sequence. In contrast to previous low-rank projection methods, our dynamic projection method is more flexible and robust to semantic-preserving positional variations (e.g., insertion, paraphrasing). We demonstrate that it outperforms previous low-rank methods [17, 18] on Long Range Arena benchmark [20].
- We identify a scale mismatch problem between the embeddings from the long-range and short-term attentions, and design a simple but effective dual normalization strategy, termed *DualLN*, to account for the mismatch and enhance the effectiveness of the aggregation.
- We demonstrate that Long-Short Transformer, despite its low memory and runtime complexity, outperforms the state-of-the-art models on a set of tasks from Long Range Arena, and autoregressive language modeling on enwik8 and text8. In addition, the proposed efficient attention mechanism can be easily applied to the most recent vision transformer architectures [6, 11] and provides state-of-the-art results, while being more scalable to high-resolution images. We also investigate the robustness properties of the Transformer-LS on diverse ImageNet datasets.

2 Related Work

2.1 Efficient Transformers

In recent years, many methods have been introduced for dealing with the quadratic cost of full attention. In general, they can be categorized as follows: *i)* Sparse attention mechanism with predefined patterns (e.g., sliding window), including Sparse Transformer [12], Image Transformer [13], Axial Transformer [25] for modeling images, and Longformer [14], blockwise self-attention [26], ETC [15], Big Bird [16] for modeling language. *ii)* Low-rank projection attention, including Linformer [17], Nyströmformer [18], Synthesizer [19]. For example, Linformer uses linear layers to project the original high resolution keys (K) and values (V) with length n to low resolution with size r ($r \ll n$) and allows all query tokens (Q) to attend these compressed representations. *iii)* Memory-based mechanisms like Compressive Transformer [10] and Set Transformer [27], which use extra memories for caching global long-range information for use in computing attention between distant tokens. *iv)* Kernel-based approximation of the attention matrix, including Performer [28], Linear

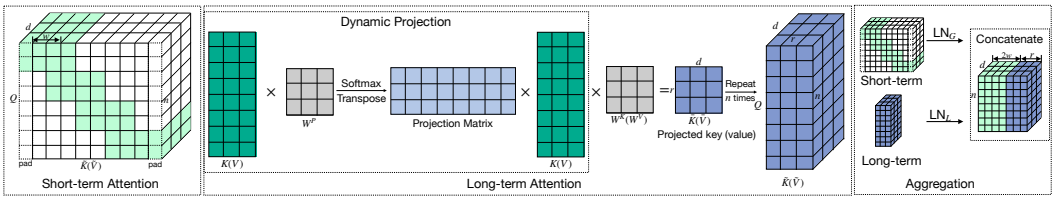

Figure 1: Long-short term attention of a single attention head. Here, the sequence length $n = 8$, hidden dimension $d = 3$, local window segment size $w = 2$, and rank of dynamic projection $r = 3$. Within the figure, $K(V)$ denotes key K or value V. In the left figure, we virtually replicate K or $V \in \mathbb{R}^{n \times d}$ into n rows, and highlight the keys and values within the attention span (denoted as $\tilde{K}(\tilde{V})$) of all n queries Q for the short-term attention. In the middle figure, all queries attend to the same projected keys $\bar{K}$ and values $\bar{V}$ within the long-term attention. In the right figure, $\tilde{K}(\tilde{V})$ and $\bar{K}(\bar{V})$ are first normalized with two sets of LayerNorms, and the queries attend to normalized $\tilde{K}(\tilde{V})$ and $\bar{K}(\bar{V})$ within their attention span simultaneously.

Transformer [29], and Random Feature Attention [30]. *vi)* Similarity and clustering based methods, including Reformer [31], Routing Transformer [32], and Sinkhorn Transformer [33].

Our method seamlessly integrates both low-rank projection and local window attentions, to leverage their strengths for modeling long-range and short-term correlations. In particular, our long-range attention uses a dynamic low-rank projection to encode the input sequence, and outperforms the previous low-rank projection method used by the Linformer [17]. In the similar vein, a few other methods also try to combine the strengths of different methods. For example, Longformer [14] and ETC [15] augment local window attention with task motivated global tokens. Such global tokens may not be applicable for some tasks (e.g., autoregressive modelling). BigBird [16] further combines local window and global token attention with random sparse attention. It is not applicable in autoregressive tasks because the global token and random sparse pattern are introduced. To compress the model footprint on edge devices, Lite Transformer [34] combines convolution and self-attention, but it still has quadratic complexity for long sequences.

2.2 Vision Transformers

Vision Transformer (ViT) [4] splits images as small patches and treats the patches as the input word tokens. It uses a standard transformer for image classification and has shown to outperform convolutional neural networks (e.g., ResNet [35]) with sufficient training data. DeiT [36] has applied the teacher-student strategy to alleviate the data efficiency problem of ViT and has shown strong comparable performance using only the standard ImageNet dataset [37]. Instead of applying transformer at a single low resolution of patches (e.g., 16×16 patches), very recent works, including Pyramid Vision Transformer (PVT) [5], Swin-Transformer [38], T2T-ViT [39], Vision Longformer (ViL) [11] and Convolutional Vision Transformer (CvT) [6], stack a pyramid of ViTs to form a multi-scale architecture and model long sequences of image patches at much higher resolution (e.g., $56 \times 56 = 3136$ patches for images with 224×224 pixels). Most of these methods have quadratic complexity of self-attention with respect to the input image size.

To reduce the complexity, Swin-Transformer [38] achieves linear complexity by limiting the computation of self-attention only within each local window. HaloNet [40] applies local attention on blocked images and only has quadratic complexity with respect to the size of the block. Perceiver [41] uses cross-attention between data and latent arrays to replace the self-attention on data to remove the quadratic complexity bottleneck. Vision Longformer (ViL) [11], another concurrent work, achieves linear complexity by adapting Longformer [14] to Vision. ViL augments local window attention with task-specific global tokens, but the global tokens are not applicable for decoding task (e.g., image synthesis [23, 24]). In contrast, our method reduces the quadratic cost to linear cost by combining local window attention with global dynamic projection attention, which can be applied to both encoding and decoding tasks.

3 Long-Short Transformer

Transformer-LS approximates the full attention by aggregating long-range and short-term attentions, while maintaining its ability to capture correlations between all input tokens. In this section, we first

introduce the preliminaries of multi-head attention in Transformer. Then, we present the short-term attention via sliding window, and long-range attention via dynamic projection, respectively. After that, we propose the aggregating method and dual normalization (DualLN) strategy. See Figure 1 for an illustration of our long-short term attention.

3.1 Preliminaries and Notations

Multi-head attention is a core component of the Transformer [1], which computes contextual representations for each token by attending to the whole input sequence at different representation subspaces. It is defined as

$$\text{MultiHead}(Q, K, V) = \text{Concat}(H_1, H_2, ..., H_h)W^O, \tag{1}$$

where $Q, K, V \in \mathbb{R}^{n\times d}$ are the query, key and value embeddings, $W^O \in \mathbb{R}^{d\times d}$ is the projection matrix for output, the i-th head $H_i \in \mathbb{R}^{n\times d_k}$ is the scaled dot-product attention, and $d_k = d/h$ is the embedding dimension of each head,

$$H_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) = \text{softmax}\left[\frac{QW_i^Q\left(KW_i^K\right)^\intercal}{\sqrt{d_k}}\right]VW_i^V = A_iVW_i^V, \tag{2}$$

where $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d\times d_k}$ are learned projection matrices, and $A_i \in \mathbb{R}^{n\times n}$ denotes the full attention matrix for each attention head. The complexity of computing and storing A_i is $O(n^2)$, which can be prohibitive when n is large. For simplicity, our discussion below is based on the case of 1D input sequences. It is straightforward to extend to the 2D image data given a predetermined order.

3.2 Short-term Attention via Segment-wise Sliding Window

We use the simple yet effective sliding window attention to capture fine-grained local correlations, where each query attends to nearby tokens within a fixed-size neighborhood. Similar techniques have also been adopted in [14, 16, 11]. Specifically, we divide the input sequence into disjoint segments with length w for efficiency reason. All tokens within a segment attend to all tokens within its home segment, as well as $w/2$ consecutive tokens on the left and right side of its home segment (zero-padding when necessary), resulting in an attention span over a total of $2w$ key-value pairs. See Figure 5 in Appendix for an illustration. For each query Q_t at the position t within the i-th head, we denote the $2w$ key-value pairs within its window as $\tilde{K}_t, \tilde{V}_t \in \mathbb{R}^{2w\times d}$. For implementation with PyTorch, this segment-wise sliding window attention is faster than the per-token sliding window attention where each token attends to itself and w tokens to its left and right, and its memory consumption scales linearly with sequence length; see [14] and our Figure 3 for more details.

The sliding window attention can be augmented to capture long-range correlations in part, by introducing different dilations to different heads of sliding window attention [14]. However, the dilation configurations for different heads need further tuning and an efficient implementation of multi-head attention with different dilations is non-trivial. A more efficient alternative is to augment sliding window attention with random sparse attention [16], but this does not guarantee that the long-range correlations are captured in each layer as in full attention. In the following section, we propose our long-range attention to address this issue.

3.3 Long-range Attention via Dynamic Projections

Previous works have shown that the self-attention matrix can be well approximated by the product of low-rank matrices [17]. By replacing the full attention with the product of low-rank matrices [42, 19, 18, 43, 28], each query is able to attend to all tokens. Linformer [17] is one of the most representative models in this category. It learns a fixed projection matrix to reduce the length of the keys and values, but the fixed projection is inflexible to semantic-preserving positional variations.

Starting from these observations, we parameterize the dynamic low-rank projection at i-th head as $P_i = f(K) \in \mathbb{R}^{n\times r}$, where $r \ll n$ is the low rank size and P_i depends on all the keys $K \in \mathbb{R}^{n\times d}$ of input sequence. It projects the $(n \times d_k)$-dimensional key embeddings KW_i^K and value embeddings VW_i^V into shorter, $(r \times d_k)$-dimensional key $\bar{K}_i$ and value $\bar{V}_i$ embeddings. Unlike Linformer [17], the low-rank projection matrix is dynamic, which depends on the input sequence and is intended to be more flexible and robust to, e.g., insertion, deletion, paraphrasing, and other operations that change sequence length. See Table 2 for examples. Note that, the query embeddings $QW_i^Q \in \mathbb{R}^{n\times d_k}$ are kept

at the same length, and we let each query attend to $\bar{K}_i$ and $\bar{V}_i$. In this way, the full $(n \times n)$ attention matrix can be decomposed into the product of two matrices with r columns or rows. Specifically, we define the dynamic projection matrix $P_i \in \mathbb{R}^{n \times r}$ and the key-value embeddings $\bar{K}_i, \bar{V}_i \in \mathbb{R}^{r \times d_k}$ of low-rank attention as

$$P_i = \text{softmax}(KW_i^P), \ \bar{K}_i = P_i^\intercal K W_i^K, \bar{V}_i = P_i^\intercal V W_i^V, \tag{3}$$

where $W_i^P \in \mathbb{R}^{d \times r}$ are learnable parameters,[2] and the softmax normalizes the projection weights on the first dimension over all n tokens, which stabilizes training in our experiments. Note that $K = V$ in all the experiments we have considered, so P_i remains the same if it depends on V. The computational complexity of Eq. 3 is $O(rn)$.

To see how the full attention is replaced by the product of low-rank matrices, we compute each head $H_i \in \mathbb{R}^{n \times d_k}$ of long-range attention as,

$$\bar{H}_i = \underbrace{\text{softmax}\left[\frac{QW_i^Q \bar{K}_i^\intercal}{\sqrt{d_k}}\right]}_{\bar{A}_i} \bar{V}_i = \bar{A}_i\big(P_i^\intercal V W_i^V\big), \tag{4}$$

so the full attention is now replaced with the implicit product of two low-rank matrices $\bar{A}_i \in \mathbb{R}^{n \times r}$ and $P_i^\intercal \in \mathbb{R}^{r \times n}$, and the computational complexity is reduced to $O(rn)$. Note the effective attention weights of a query on all tokens still sum to 1. Our global attention allows each query to attend to all token embeddings within the same self-attention layer. In contrast, the sparse attention mechanisms [14, 16] need stack multiple layers to build such correlations.

Application to Autoregressive Models: In autoregressive models, each token can only attend to the previous tokens, so the long-range attention should have a different range for different tokens. A straightforward way to implement our global attention is to update $\bar{K}_i, \bar{V}_i$ for each query recurrently, but this requires re-computing the projection in Eq. (3) for every token due to the nonlinearity of softmax, which results in $O(rn^2)$ computational complexity. To preserve the linear complexity, for autoregressive models, we first divide the input sequence into equal-length segments with length l, and apply our dynamic projection to extract $\bar{K}_i, \bar{V}_i$ from each segment. Each token can only attend to $\bar{K}_i, \bar{V}_i$ of segments that do not contain its future tokens. Formally, let Q_t be the query at position t, $K_{(l-1)s:ls}, V_{(l-1)s:ls}$ be the key-value pairs from the s-th segment, and $s_t = \lfloor t/l \rfloor$. For autoregressive models, we compute the long-range attention of Q_t by attending to $K_{i,t}, V_{i,t}$, defined as

$$\bar{K}_{i,t} = [P_{i,1}^\intercal K_{1:l}; ...; P_{i,s_t}^\intercal K_{(l-1)s_t:ls_t}] W_i^K, \bar{V}_{i,t} = [P_{i,1}^\intercal V_{1:l}; ...; P_{i,s_t}^\intercal V_{(l-1)s_t:ls_t}] W_i^V. \tag{5}$$

In this way, the dynamic low-rank projection is applied to each segment only once in parallel, preserving the linear complexity and the high training speed. By comparison, Random Feature Attention [30] is slow at training due to the requirement for recurrence.

3.4 Aggregating Long-range and Short-term Attentions

To aggregate the local and long-range attentions, instead of adopting different attention mechanisms for different heads [12, 14, 34], we let each query at i-th head attend to the union of keys and values from the local window and global low-rank projections, thus it can learn to select important information from either of them. We find this aggregation strategy works better than separating the heads in our initial trials with the autoregressive language models. Specifically, for the i-th head, we denote the global low-rank projected keys and values as $\bar{K}_i, \bar{V}_i \in \mathbb{R}^{r \times d_k}$, and the local keys and values as $\tilde{K}_t, \tilde{V}_t \in \mathbb{R}^{2w \times d}$ within the local window of position t for the query Q_t. Then the i-th attention $H_{i,t}$ at position t is

$$H_{i,t} = \text{softmax}\left[\frac{Q_t W_i^Q \left[\tilde{K}_t W_i^K; \bar{K}_i\right]^\intercal}{\sqrt{d_k}}\right] [\tilde{V}_t W_i^V; \bar{V}_i]. \tag{6}$$

where $[\cdot\,;\cdot]$ denotes concatenating the matrices along the first dimension. Furthermore, we find a scale mismatch between the initial norms of $\tilde{K}_t W_i^K$ and $\bar{K}_i$, which biases the attention to the local window at initialization for both language and vision tasks. We introduce a normalization strategy (DualLN) to align the norms and improve the effectiveness of the aggregation in the following.

[2] For the CvT-based vision transformer model, we replace W_i^P with a depth-wise separable convolution, just as its query, key and value projections.

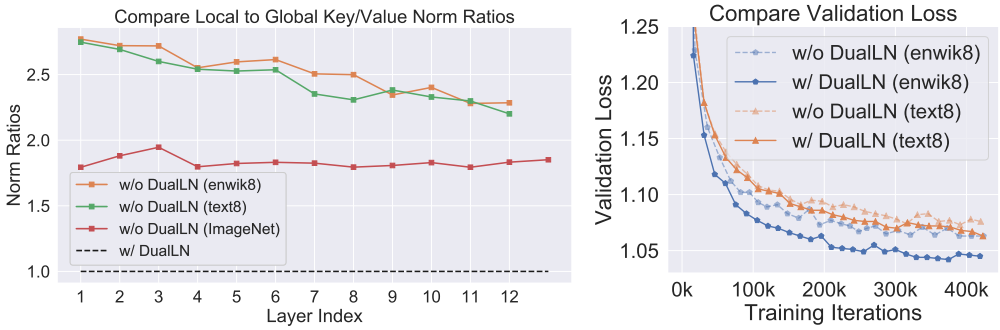

Figure 2: Left: Ratios of the average ℓ_2 norms of the local window to global low-rank key/value embeddings at initialization. Without DualLN, the sparse and low-rank embeddings have a magnitude mismatch. With DualLN, the ratios will be 1.0 at every layer, which will facilitate optimization. Right: The validation loss of Transformer-LS with and without DualLN on enwik8 and text8.

DualLN: For Transformers with Layer Normalization (LN) (see [44] for an illustration), the K_i, V_i embeddings are the outputs of LN layers, so they have zero mean and unit variance at initialization. The ℓ_2 norm of vectors with zero-mean entries is proportional to their variance in expectation. We note a weighted average will reduce the variance and therefore the norm of such zero-mean vectors. As a result, the embedding vectors from low-rank attention in the weighted average $\bar{K}_i, \bar{V}_i$ of Eq. (3) will have smaller norms than the regular key and value embeddings from sliding window attention (see Figure 2 Left for an illustration). This scale mismatch causes two side effects. First, the inner product $Q_t W_i^Q \bar{K}_i^\intercal$ from local-rank component tends to have smaller magnitude than the local window one, thus the attention scores on long-range attention is systematically smaller. Second, the key-value pairs $\bar{K}_i, \bar{V}_i$ for the low-rank attention will naturally have less impact on the direction of H_i even when low-rank and local window are assigned with same attention scores, since $\bar{V}_i$ has smaller norms. Both effects lead to small gradients on the low-rank components and hinders the model from learning to effectively use the long-range correlations.

To avoid such issues, we add two sets of Layer Normalizations after the key and value projections for the local window and global low-rank attentions, so that their scales are aligned at initialization, but the network can still learn to re-weight the norms after training. Specifically, the aggregated attention is now computed as

$$H_{i,t} = \text{softmax}\left[\frac{Q_t W_i^Q \left[\text{LN}_L(\tilde{K}_t W_i^K); \text{LN}_G(\bar{K}_i)\right]^\intercal}{\sqrt{d_k}}\right] [\text{LN}_L(\tilde{V}_t W_i^V); \text{LN}_G(\bar{V}_i)], \tag{7}$$

where $\text{LN}_L(\cdot), \text{LN}_G(\cdot)$ denote the Layer Normalizations for the local and global attentions respectively. In practice, to maintain the consistency between the local attention and dynamic projection, we use $\text{LN}_L(K), \text{LN}_L(V)$ instead of K, V to compute $\bar{K}_i, \bar{V}_i$ in Eq. 3. As illustrated in Figure 2 Right, the Transformer-LS models trained with DualLN has consistently lower validation loss than the models without DualLN.

4 Experiments

In this section, we demonstrate the effectiveness and efficiency of our method in both language and vision domains. We use PyTorch for implementation and count the FLOPs using fvcore [45].

4.1 Bidirectional Modeling on Long Range Arena and IMDb

To evaluate Long-Short Transformer as a bidirectional encoder for long text, we train our models on the three NLP tasks, **ListOps**, **Text**, and **Retrieval**, from the recently proposed Long Range Arena (LRA) benchmark [20], following the setting of Peng et al. [30] and Tay et al. [46]. For fair comparisons, we use the PyTorch implementation and the same data preprocessing/split, training hyperparameters and model size from [18], except for **Retrieval** where we accidentally used more warmup steps and improved the results for all models. See Appendix B for more details. The results

Table 1: Accuracy (%) and FLOPs (G) on Long Range Arena (LRA), with the model configs annotated (see Table 7 for more). All results are averages of 4 runs with different random seeds.

Task (mean $\pm$ std.) of sequence length	ListOps (888 ± 339)		Text (1296 ± 893)		Retrieval (3987 ± 560)		Average
Model	Acc.	FLOPs	Acc.	FLOPs	Acc.	FLOPs	Acc.
Full Attention [1]	37.13	1.21	65.35	4.57	**82.30**	9.14	61.59
Reformer [31] (2)	36.44	0.27	64.88	0.58	78.64	1.15	59.99
Linformer [17] (k=256)	37.38	0.41	56.12	0.81	79.37	1.62	57.62
Performer [28] ($r = 256$)	32.78	0.41	65.21	0.82	81.70	1.63	59.90
Nyströmformer [18] ($l = 128$)	37.34	0.61	65.75	1.02	81.29	2.03	61.46
Transformer-LS ($w, r = 8, 32$)	**37.50**	0.20	**66.01**	0.40	81.79	0.80	**61.77**
Dynamic Projection (best)	37.79	0.15	66.28	0.69	81.86	2.17	61.98
Transformer-LS (best)	**38.36**	0.16	**68.40**	0.29	**81.95**	2.17	**62.90**

Table 2: Comparing the robustness of the models under test-time insertions and deletions. DP refers to long-range attention via Dynamic Projection, and Win. refers to sliding window attention.

Task	Text			Retrieval		
Test Perturb	None	Insertion	Deletion	None	Insertion	Deletion
Linformer	56.12	55.94	54.91	79.37	53.66	51.75
DP	66.28	63.16	58.95	81.86	**70.01**	**64.98**
Linformer + Win.	59.63	56.69	56.29	79.68	52.83	52.13
DP + Win. (ours)	**68.40**	**66.34**	**62.62**	**81.95**	69.93	64.19

Table 3: Comparing the results of pretrained language models fine-tuned on IMDb.

Model	RoBERTa-base	RoBERTa-large	Longformer-base	LS-base	LS-large
Accuracy	95.3	96.5	95.7	96.0	**96.8**

on these three tasks are given in Table 1. Results of the other two image-based tasks of LRA, as well as models implemented in JAX, are given in Appendix C and C.2.

In addition, we follow the pretraining procedure of Longformer [14] to pretrain our models based on RoBERTa-base and RoBERTa-large [47], and fine-tune it on the IMDb sentiment classification dataset. The results are given in Table 3.

Results. From Table 3, our base model outperforms Longformer-base, and our large model achieves improvements over RoBERTa-large, demonstrating the benefits of learning to model long sequences. Comparisons with models on LRA are given in Table 1. Transformer-LS (best) with the best configurations of w, r for each task are given in Table 7 in Appendix B. We also report the results of using fixed hyperparameter $w = 8, r = 32$ on all tasks. Overall, our Transformer-LS (best) is significantly better than other efficient Transformers, and the model with $w, r = 8, 32$ performs favorably while using only about 50% to 70% computation compared to other efficient Transformers on all three tasks. The advantage of aggregating local and long-range attentions is the most significant on **ListOps**, which requires the model to understand the tree structures involving both long-term and short-term relations. On **Retrieval**, where document-level encoding capability is tested, we find our global attention more effective than window attention. The test accuracy of using only dynamic projection is about 10% higher than Linformer on **Text** (i.e., 66.28 vs. 56.12), which has the highest variance in sequence length (i.e. standard deviation 893). This demonstrates the improved flexibility of dynamic projection at learning representations for data with high variance in sequence length, compared to the learned but fixed projection of Linformer. Similarly, Linformer, Nyströmformer and our model outperform full attention on **ListOps**, indicating they may have better inductive bias, and efficient Transformers can have better efficacy beyond efficiency.

Robustness of Dynamic Projection. In Table 2, we compare the robustness of Linformer and the proposed Dynamic Projection (DP) against insertion and deletion on Text and Retrieval tasks of LRA. We train the models on the original, clean training sets and only perturb their test sets. For insertion, we insert 10 random punctuations at 10 random locations of each test sample. For deletion, we delete

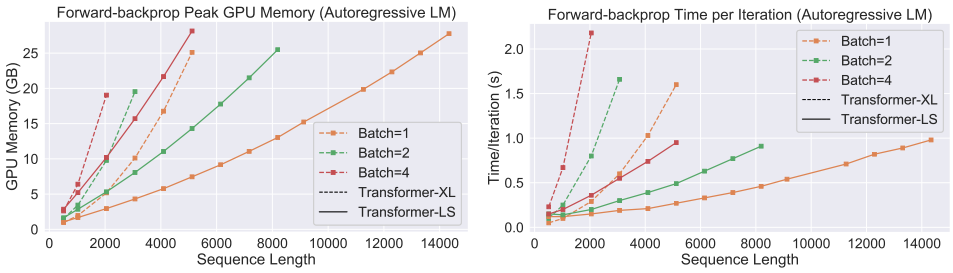

Figure 3: Running time and memory consumption of Transformer-XL (full attention) and our Transformer-LS on Char-LM. We increase the sequence length until we use up the 32GB of memory on a V100 GPU. Transformer-LS is the same smaller model in Table 4. We use dashed lines to represent the full attention Transformer and solid lines to represent our model. We use different colors to represent different batch sizes.

Table 4: BPC (↓) of smaller models on enwik8 and text8 (left), and larger models on enwik8 (right).

Method	#Param	text8		enwik8	
		Dev	Test	Dev	Test
T12 [49]	44M	-	1.18	-	1.11
Transformer-XL [9]	41M	-	-	-	1.06
Reformer [31]	-	-	-	-	1.05
Adaptive [50]	38M	1.05	1.11	1.04	1.02
BP-Transformer [51]	38M	-	1.11	-	1.02
Longformer [20]	41M	1.04	1.10	1.02	1.00
Transformer-LS	44M	1.03	**1.09**	1.01	**0.99**

Method	#Param	Test BPC
Transformer-XL [9]	88M	1.03
Transformer-XL [9]	277M	0.99
Routing [32]	223M	0.99
Longformer [14]	102M	0.99
Sparse [12]	95M	0.99
Adaptive [50]	209M	0.98
Compressive [10]	227M	**0.97**
Transformer-LS	110M	**0.97**

all punctuations from the test samples. Both transforms are label-preserving in most cases. By design, dynamic projection is more robust against location changes.

4.2 Autoregressive Language Modeling

We compare our method with other efficient transformers on the character-level language modeling where each input token is a character.

Setup. We train and evaluate our model on enwik8 and text8, each with 100M characters and are divided into 90M, 5M, 5M for train, dev, test, following [48]. Our smaller 12-layer and larger 30-layer models are Pre-LN Transformers with the same width and depth as Longformer [20], except that we add relative position encoding to the projected segments in each layer. We adopt the cache mechanism of Transformer-XL [9], setting the cache size to be the same as the input sequence length. We follow similar training schedule as Longformer, and train our model in 3 phases with increasing sequence lengths. The input sequence lengths are 2048, 4096 and 8192 respectively for the 3 phases. By comparison, Longformer trains their model in 5 phases on GPUs with 48GB memory (The maximal of ours is 32GB) where the sequence length is 23,040 in the last phase. The window size of Longformer increases with depth and its average window size is 4352 in phase 5, while our effective number of attended tokens is 1280 on average in the last phase. Each experiment takes around 8 days to finish on 8 V100 GPUs. Detailed hyperparameters are shown in Appendix D. For testing, same as Longformer, we split the dataset into overlapping sequences of length 32K at a step size of 512, and evaluate the BPCs for predicting the next 512 tokens given the previous 32K characters.

Results Table 4 shows comparisons on text8 and enwik8. Our method has achieved state-of-the-art results. On text8, we achieve a test BPC of 1.09 with the smaller model. On enwik8, our smaller model achieves a test BPC of 0.99, and outperforms the state-of-the-art models with comparable number of parameters. Our larger model obtains a test BPC of 0.97, on par with the Compressive Transformer with $2\times$ parameters. Our results are consistently better than Longformer which is trained on longer sequences with 5 stages and 48 GPU memory. In Figure 3, we show our model is much more memory and computational efficient than full attention.

Table 5: Test accuracies on ImageNet, ImageNet Real [52], and ImageNet V2 [53] of models trained on ImageNet-1K. **Grey-colored rows are our results**. CvT*-LS denotes our long-short term attention based on the non-official CvT implementation. ViL models with LS suffixes are our long-short term attention based on the official ViL implementation with relative positional bias. We also provide the latency of models tested using batch size 32 on the same V100 GPU. Our improvements over ViL is mainly from a better implementation of the short-term attention.

Model	#Param (M)	Image Size	FLOPs (G)	ImageNet top-1 (%)	Real top-1 (%)	V2 top-1 (%)	Latency (s)
ResNet-50	25	224^2	4.1	76.2	82.5	63.3	-
ResNet-101	45	224^2	7.9	77.4	83.7	65.7	-
ResNet-152	60	224^2	11	78.3	84.1	67.0	-
DeiT-S [36]	22	224^2	4.6	79.8	85.7	68.5	-
DeiT-B [36]	86	224^2	17.6	81.8	86.7	70.9	-
PVT-Medium [5]	44	224^2	6.7	81.2	-	-	-
PVT-Large [5]	61	224^2	9.8	81.7	-	-	-
Swin-S [38]	50	224^2	8.7	83.2	-	-	-
Swin-B [38]	88	224^2	15.4	83.5	-	-	0.115
PVTv2-B4 [54]	62.6	224^2	10.1	83.6	-	-	-
PVTv2-B5 [54]	82.0	224^2	11.8	83.8	-	-	-
ViT-B/16 [4]	86	384^2	55.5	77.9	-	-	-
ViT-L/16 [4]	307	384^2	191.1	76.5	-	-	-
DeiT-B [36]	86	384^2	55.5	83.1	-	-	-
Swin-B [38]	88	384^2	47.1	**84.5**	-	-	0.378
CvT-13 [6]	20	224^2	6.7	81.6	86.7	70.4	0.122
CvT-21 [6]	32	224^2	10.1	82.5	87.2	71.3	0.165
CvT*-LS-13	20.3	224^2	4.9	81.9	87.0	70.5	0.083
CvT*-LS-17	23.7	224^2	9.8	82.5	87.2	71.6	-
CvT*-LS-21	32.1	224^2	7.9	82.7	87.5	71.9	0.122
CvT*-LS-21S	30.1	224^2	11.3	**82.9**	87.4	71.7	-
CvT-13 [6]	20	384^2	31.9	83.0	87.9	71.9	-
CvT-21 [6]	32	384^2	45.0	83.3	87.7	71.9	-
CvT*-LS-21	32.1	384^2	23.9	83.2	88.0	72.5	-
CvT*-LS-21	32.1	448^2	34.2	**83.6**	88.2	72.9	-
ViL-Small [14]	24.6	224^2	4.9	82.4	-	-	-
ViL-Medium [14]	39.7	224^2	8.7	83.5	-	-	0.106
ViL-Base [14]	55.7	224^2	13.4	83.7	-	-	0.164
ViL-LS-Medium	39.8	224^2	8.7	83.8	-	-	0.075
ViL-LS-Base	55.8	224^2	13.4	**84.1**	-	-	0.113
ViL-LS-Medium	39.9	384^2	28.7	**84.4**	-	-	0.271

4.3 ImageNet Classification

We train and evaluate the models on ImageNet-1K with 1.3M images and 1K classes. We use CvT [6] and ViL [11], state-of-the art vision transformer architectures, as the backbones and replace their attention mechanisms with our long-short term attention, denoted as CvT*-LS and ViL-`size`-LS in Table 5. CvT uses overlapping convolutions to extract dense patch embeddings from the input images and feature maps, resulting in a long sequence length in the early stages (e.g., $56 \times 56 = 3136$ patches for images with 224^2 pixels). For ViL, our sliding window uses the same group size w, but each token attends to at most $2w \times 2w$ (rounding when necessary) tokens inside the window, instead of $3w \times 3w$ as ViL, which allows adding our dynamic projection without increasing the FLOPs. We set $r = 8$ for the dynamic projections for both ViL-LS-Medium and ViL-LS-Base. Note that, our efficient attention mechanism does not depend on the particular architecture, and it can be applied to other vision transformers [e.g., 4, 36, 5]. Please refer to Appendix E for more details.

Classification Results. The results are shown in the Table 5, where we also list test accuracies on ImageNet Real and ImageNet V2. Except for CvT, we compare with the original ViT [4] and the enhanced DeiT [36], PVT [5] that also uses multi-scale stragey, ViL [11] that uses window attention and global tokens to improve the efficiency. Training at high-resolution usually improves the test

Table 6: Robustness evaluation on various ImageNet datasets. Top-1/Acc.: Top-1 accuracy. mCE: Mean Corruption Error. Mixed-same/Mixed-rand: accuracies on MIXED-SAME/MIXED-RAND subsets.

Model	Params	ImageNet	IN-C [56]	IN-A [57]	IN-R [58]	ImageNet-9 [59]	
	(M)	Top-1	mCE (↓)	Acc.	Acc.	Mixed-same	Mixed-rand
ResNet-50 [35]	25.6	76.2	78.9	6.2	35.3	87.1	81.6
DeiT-S [36]	22.1	79.8	57.1	19.0	41.9	89.1	84.2
CvT-13	20	81.6	59.6	25.4	42.9	90.5	85.7
CvT-21	32	82.5	56.2	**31.1**	42.6	90.5	85.0
CvT*-LS-13	20.3	81.9	58.7	27.0	42.6	90.7	85.6
CvT*-LS-21	32.1	**82.7**	**55.2**	29.3	**45.0**	**91.5**	**85.8**

accuracy of vision transformer. With our long-short term attention, we can easily scale the training to higher resolution, and the performance of CvT*-LS and ViL-LS also improves. Our best model with CvT (CvT*-LS-21 at 448^2) achieves 0.3% higher accuracy than the best reported result of CvT while using the same amount of parameters and 76% of its FLOPs. In CvT architecture, the spatial dimension of feature maps in earlier stages are large, representing more fine-grained details of the image. Similar to training with high-resolution images, the model should also benefit from denser feature maps. With our efficient long-short term attention, we can better utilize these fine-grained feature maps with less concerns about the computational budget. In this way, our CvT*-LS-17 achieves better result than CvT-21 at resolution 224 using fewer parameters and FLOPs, and our CvT*-LS-21S model further improves our CvT*-LS-21 model.

Our ViL-LS-Medium and ViL-LS-Base with long-short term attention improve the accuracies of ViL-Medium and ViL-Base from 83.5 and 83.7 to 83.8 and 84.1 respectively, without an increase in FLOPs. When increasing the resolution for training ViL-LS-Medium from 224^2 to 384^2, the FLOPs increased (approximately) linearly and the accuracy improved by 0.6%, showing our method still benefits greatly from increased resolution while maintaining the linear complexity in practice.

Short-term Attention Suppresses Oversmoothing. By restricting tokens from different segments to attend to different windows, our short-term sparse local attention encourages diversity of the feature representations and helps to alleviate the over-smoothing problem [55] (where all queries extract similar information in deeper layers and the attention mechanism is less important), thus can fully utilize the depth of the network. As in [55], we provide the cosine similarity of patch embeddings of our CvT*-LS-13 and re-implemented CvT-13 (81.1 accuracy) in Figure 6 within Appendix. This is one of the reasons why our efficient attention mechanism can get even better results than the full attention CvT model in the same setting.

Robustness evaluation on Diverse ImageNet Datasets. As vision models have been widely used in safety-critical applications (e.g. autonomous driving), their robustness is vital. In addition to out-of-distribution robustness (ImageNet-Real and Imageet-v2), we further investigate the robustness of our vision transformer against common corruption (ImageNet-C), semantic shifts (ImageNet-R), Background dependence (ImageNet-9) and natural adversarial examples (ImageNet-A). We compare our methods with standard classification methods, including CNN-based model (ResNet [35]) and Transformer-based models (DeiT [36]) with similar numbers of parameters. As shown in Table 6, we observe that our method significantly outperforms the CNN-based method (ResNet-50). Compared to DeiT, our models also achieve favorable improvements. These results indicate that the design of different attention mechanisms plays an important role for model robustness, which sheds new light on the design of robust vision transformers. More details and results can be found in Appendix E.

5 Conclusion

In this paper, we introduced Long-Short Transformer, an efficient transformer for long sequence modeling for both language and vision domain, including both bidirectional and autoregressive models. We design a novel global attention mechanism with linear computational and memory complexity in sequence length based on a dynamic projection. We identify the scale mismatch issue and propose the DualLN technique to eliminate the mismatch at initialization and more effectively aggregate the local and global attentions. We demonstrate that our method obtains the state-of-the-art results on the Long Range Arena, char-level language modeling and ImageNet classification. We look forward to extending our methods to more domains, including document QA, object detection and semantic segmentation on high-resolution images.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, volume 30, 2017.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, 2019.

[3] 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.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[5] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021.

[6] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021.

[7] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *TACL*, 7:453–466, 2019.

[8] Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. Hierarchical transformers for long document classification. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 838–844. IEEE, 2019.

[9] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, 2019.

[10] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive Transformers for long-range sequence modelling. In *ICLR*, 2020.

[11] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. *arXiv preprint arXiv:2103.15358*, 2021.

[12] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[13] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *ICML*, pages 4055–4064, 2018.

[14] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[15] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. Etc: Encoding long and structured inputs in transformers. In *EMNLP*, pages 268–284, 2020.

[16] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big Bird: Transformers for longer sequences. In *NeurIPS*, 2020.

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

[18] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. *AAAI*, 2021.

[19] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention in transformer models. In *ICML*, 2021.

[20] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long Range Arena: A benchmark for efficient transformers. In *ICLR*, 2021.

[21] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. *arXiv preprint arXiv:2103.15358*, 2021.

[22] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[23] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *arXiv preprint arXiv:1906.00446*, 2019.

[24] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021.

[25] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[26] Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. *arXiv preprint arXiv:1911.02972*, 2019.

[27] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019.

[28] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *ICLR*, 2021.

[29] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, 2020.

[30] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. *ICLR*, 2021.

[31] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020.

[32] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *TACL*, 9:53–68, 2021.

[33] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *ICML*. PMLR, 2020.

[34] Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite transformer with long-short range attention. *arXiv preprint arXiv:2004.11886*, 2020.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[36] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[37] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[38] 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. *arXiv preprint arXiv:2103.14030*, 2021.

[39] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

[40] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. *arXiv preprint arXiv:2103.12731*, 2021.

[41] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention. *arXiv preprint arXiv:2103.03206*, 2021.

[42] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*. PMLR, 2020.

[43] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. *ICLR*, 2021.

[44] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *ICML*. PMLR, 2020.

[45] fvcore: Flop counter for pytorch models. https://github.com/facebookresearch/fvcore/blob/master/docs/flop_count.md, 2021.

[46] Yi Tay, Mostafa Dehghani, Vamsi Aribandi, Jai Gupta, Philip Pham, Zhen Qin, Dara Bahri, Da-Cheng Juan, and Donald Metzler. Omninet: Omnidirectional representations from transformers. *ICML*, 2021.

[47] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[48] Matt Mahoney. Large text compression benchmark. *URL http://mattmahoney.net/dc/textdata*, 6, 2009.

[49] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. In *AAAI*, volume 33, pages 3159–3166, 2019.

[50] Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers. *ACL*, 2019.

[51] Zihao Ye, Qipeng Guo, Quan Gan, Xipeng Qiu, and Zheng Zhang. Bp-transformer: Modelling long-range context via binary partitioning. *arXiv preprint arXiv:1911.04070*, 2019.

[52] Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020.

[53] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019.

[54] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvtv2: Improved baselines with pyramid vision transformer. 2021.

[55] Chengyue Gong, Dilin Wang, Meng Li, Vikas Chandra, and Qiang Liu. Improve vision transformers training by suppressing over-smoothing. *arXiv preprint arXiv:2104.12753*, 2021.

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

[57] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *arXiv preprint arXiv:1907.07174*, 2019.

[58] 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. *arXiv preprint arXiv:2006.16241*, 2020.

[59] Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. *ArXiv preprint arXiv:2006.09994*, 2020.

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *CVPR*, 2015.

[61] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[62] Nikita Nangia and Samuel Bowman. Listops: A diagnostic dataset for latent tree learning. In *NAACL: Student Research Workshop*, pages 92–99, 2018.

[63] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *ACL*, 2011.

[64] Dragomir R Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara. The acl anthology network corpus. *Language Resources and Evaluation*, 47(4):919–944, 2013.

[65] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[66] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[67] Lei Huang, Xianglong Liu, Yang Liu, Bo Lang, and Dacheng Tao. Centered weight normalization in accelerating training of deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2803–2811, 2017.

[68] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Micro-batch training with batch-channel normalization and weight standardization. *arXiv preprint arXiv:1903.10520*, 2019.

Long-Short Transformer: Efficient Transformers for Language and Vision (Appendix)

A Details of Norm Comparisons

As we have shown in Figure 2, the norms of the key-value embeddings from the long-term and short-term attentions, $\bar{K}, \bar{V}$ and $\tilde{K}, \tilde{V}$, are different at initialization, and the norms of $\tilde{K}, \tilde{V}$ is always larger than $\bar{K}, \bar{V}$ on different networks and datasets we have evaluated. Here, we give an explanation.

Intuitively, at initialization, following similar assumptions as [60, 61], the entries of K, V should have zero mean. Since each entry of $\bar{K}, \bar{V}$ is a weighted mean of K, V, they have smaller variance unless one of the weights is 1. Given that $\bar{K}, \bar{V}$ are also zero-mean, the norm of their embedding vectors (their rows), which is proportional to the variance, is smaller. For the key-value embeddings from short-term attention, $\tilde{K}, \tilde{V}$ are just a subset of K, V, so their embedding vectors should have the same norm as rows of K, V in expectation. Therefore, the norms of embedding vectors from $\bar{K}, \bar{V}$ will be smaller than those from $\tilde{K}, \tilde{V}$ in expectation.

B Details for Experiments on Long Range Arena

The tasks. We compare our method with the following three tasks:

- **ListOps**. ListOps [62] is designed to measure the parsing ability of models through hierarchically structured data. We follow the setting in [20] in which each instance contains 500-2000 tokens.
- **Text**. This is a binary sentiment classification task of predicting whether a movie review from IMDb is positive or negative [63]. Making correct predictions requires a model to reason with compositional unsegmented char-level long sequences with a maximum length of 4k.
- **Retrieval**. This task is based on the ACL Anthology Network dataset [64]. The model needs to classify whether there is a common citation between a pair of papers, which evaluates the model's ability to encode long sequences for similarity-based matching. The max sequence length for each byte-level document is 4k and the model processes two documents in parallel each time.

Architecture. On all tasks, the models have 2 layers, with embedding dimension $d = 64$, head number $h = 2$, FFN hidden dimension 128, smaller than those from [20]. Same as [20], we add a CLS token as a global token and use its embedding in the last layer for classification. We re-implement the methods evaluated by Xiong et al. [18], and report the best results of our re-implementation and those reported by Xiong et al. [18]. For our method, the results we run a grid search on the window size w and the projected dimension r, and keep $2w + r \leq 256$ to make the complexity similar to the other methods. The maximum sequence length for **ListOps** and **Text** are 2048 and 4096. For **Retrieval**, we set the max sequence for each of the two documents to 4096.

Table 7: Configurations of our method corresponding to the best results (Transformer-LS (best)) in Table 1.

	ListOps (2k)		Text (4k)		Retrieval (4k)	
	w	r	w	r	w	r
Dynamic Projection	0	4	0	128	0	256
Transformer-LS	16	2	1	1	1	254

Hyperparameters for Training. Our hyperparameters are the same as Nyströmformer [18] unless otherwise specified. Specifically, we follow [18] and use Adam with a fixed learning rate of 10^{-4} without weight decay, batch size 32 for all tasks. The number of warmup training steps T_w and total training steps T are different due to the difference in numbers of training samples. For **Retrieval**, we accidentally found using $T_w = 8000$ rather than the default $T_w = 800$ of [18] improves the results for all models we have evaluated. See Table 8 for the configurations of each task.

Error bars. We have already provided the average of 4 runs with different random seeds in Table 1. Here we also provide the standard deviations for these experiments in Table 9.

Table 8: Training Hyperparameters for LRA tasks.

	lr	batch size	T_w	T
ListOps	10^{-4}	32	1000	5000
Text	10^{-4}	32	8000	20000
Retrieval	10^{-4}	32	8000	30000

Table 9: Accuracy (%) and its standard deviation on Long Range Arena (LRA), with the model configurations and sequence length stats (under the dataset names) annotated. All results are averages of 4 runs with different random seeds. Note that, text has the largest variance of length (i.e., 893).

	ListOps (888 ± 339)		Text (1296 ± 893)		Retrieval (3987 ± 560)		Average
Model	Acc.	FLOPs	Acc.	FLOPs	Acc.	FLOPs	Acc.
Full Attention [1]	37.1 ± 0.4	1.21	65.4 ± 0.3	4.57	**82.3** ± 0.4	9.14	61.59
Reformer [31] (2)	36.4 ± 0.4	0.27	64.9 ± 0.4	0.58	78.6 ± 0.7	1.15	59.99
Linformer [17] (k=256)	37.4 ± 0.4	0.41	56.1 ± 1.5	0.81	79.4 ± 0.9	1.62	57.62
Performer [28] ($r = 256$)	32.8 ± 9.4	0.41	65.2 ± 0.2	0.82	81.7 ± 0.2	1.63	59.90
Nyströmformer [18] ($l = 128$)	37.3 ± 0.2	0.61	65.8 ± 0.2	1.02	81.3 ± 0.3	2.03	61.46
Transformer-LS ($w, r = 8, 32$)	**37.5** ± 0.3	0.20	**66.0** ± 0.2	0.40	81.8 ± 0.3	0.80	**61.77**
Dynamic Projection (best)	37.8 ± 0.2	0.15	66.3 ± 0.7	0.69	81.9 ± 0.5	2.17	61.98
Transformer-LS (best)	**38.4** ± 0.4	0.16	**68.4** ± 0.8	0.29	**82.0** ± 0.5	2.17	**62.90**

C Additional Results on LRA

C.1 Results on the image-based tasks of LRA

We give the results of our model on the image-based tasks, implemented in PyTorch, in Table 10.

Table 10: Comparing our model (Transformer-LS) with other methods on the image-based tasks of LRA. For the results of other models, we take their highest scores from [18] and [20].

Model	Transformer-LS	Linformer	Reformer	Performer	Sparse. Trans.	Nystromformer	Full Att.
Image	45.05	38.56	43.29	42.77	44.24	41.58	42.44
Pathfinder	76.48	76.34	69.36	77.05	71.71	70.94	74.16

C.2 Compare models implemented in JAX

To compare the results with the implementations from the original LRA paper [20], we re-implement our method in JAX and give the comparisons with other methods in Table 11. The accuracies of other methods come from the LRA paper. We evaluate the per-batch latency of all models on A100 GPUs using their official JAX implementation from the LRA paper. Our method still achieves improvements while being efficient enough. We were unable to run Reformer with the latest JAX since JAX has deleted `jax.custom_transforms`, which is required by the Reformer implementation, from its API.[3] Note the relative speedups from the LRA paper are evaluated on TPUs.

D Details for Autoregressive Language Modeling

An example of long-short term attention for autoregressive models. We give an illustration for the segment-wise dynamic projection for autoregressive models as discussed in Section 3.3. With the segment-wise formulation, we can first compute the low-rank projection for each segment in parallel, and each query will only attend to the tokens from segments that do not contain the future token or the query token itself. The whole process is efficient and maintain the $O(n)$ complexity, unlike RFA [30] which causes a slow-down in training due to the requirement for cumulative sum. However,

[3] https://github.com/google/jax/pull/2026

Table 11: Comparing the test scores and latency of models on LRA, implemented in JAX.

Model	ListOps		Text		Retrieval	
	Acc.	Latency (s)	Acc.	Latency (s)	Acc.	Latency (s)
Local Att	15.82	0.151	52.98	0.037	53.39	0.142
Linear Trans.	16.13	0.156	65.9	0.037	53.09	0.142
Reformer	37.27	-	56.10	-	53.40	-
Sparse Trans.	17.07	0.447	63.58	0.069	59.59	0.273
Sinkhorn Trans.	33.67	0.618	61.20	0.048	53.83	0.241
Linformer	35.70	0.135	53.94	0.031	52.27	0.117
Performer	18.01	0.138	65.40	0.031	53.82	0.120
Synthesizer	36.99	0.251	61.68	0.077	54.67	0.306
Longformer	35.63	0.380	62.85	0.112	56.89	0.486
Transformer	36.37	0.444	64.27	0.071	57.46	0.273
BigBird	36.05	0.269	64.02	0.067	59.29	0.351
Transformer-LS	37.65	0.187	76.64	0.037	66.67	0.201

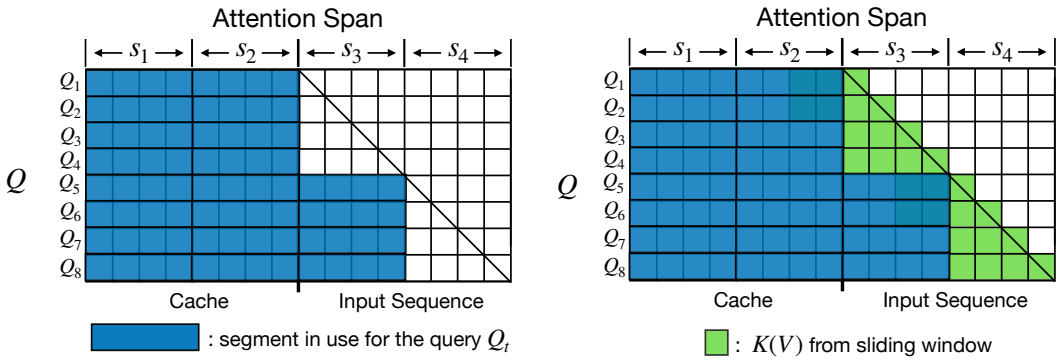

Figure 4: An illustration of effective attention span (colored regions) in Transformer-LS when the segment size for the low-rank attention is $\ell = 4$, and the segment size for the sliding window attention is $w = 2$. Left: the attention span of only the low-rank attention (segment-wise dynamic projection). Right: the attention span of the aggregated attention.

in this way, some of the most recent tokens are ignored, as shown in Figure 4 (left). The window attention (with segment size $w \geq l/2$) becomes an indispensable component in this way, since it fills the gap for the missing recent tokens, as shown in Figure 4.

Experimental Setup. Throughout training, we set the window size $w = 512$, the segment length $l = 16$, and the dimension of the dynamic low-rank projection $r = 1$, which in our initial experiments achieved better efficiency-BPC trade-off than using $l = 32, r = 1$ or $l = 64, r = 4$. Our small and large models have the same architecture as Longformer [14], except for the attention mechanisms. We use similar training schedules as Longformer [14]. Specifically, for all models and both datasets, we train the models for 430k/50k/50k steps with 10k/5k/5k linear learning rate warmup steps, and use input sequence lengths 2048/4096/8192 for the 3 phases. We use constant learning rate after warmup. We compared learning rates from {1.25e-4, 2.5e-4,5e-4,1e-3} for 100k iterations and found 2.5e-4 to work the best for both models on enwik8, and 5e-4 to work the best on text8. The batch sizes for the 3 phases are 32, 32, 16 respectively. Unlike Longformer and Transformer-XL, we remove gradient clipping and found the model to have slightly faster convergence in the beginning while converging reliably. For smaller models, we use dropout rate 0.2 and weight decay 0.01. For the larger model, we use dropout 0.4 and weight decay 0.1.

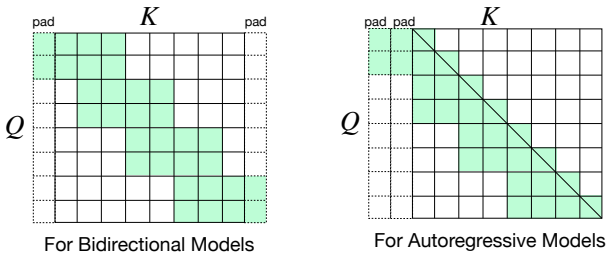

Figure 5: An illustration of our sliding window attention in 1D autoregressive and bidirectional models. Here, we use a group size $w = 2$. Each token inside each group are restricted to attend to at most $2w$ tokens. In the bidirectional model, they attend to w tokens from the home segment, and $w/2$ tokens to the left and right of the home segment respectively. In the autoregressive model, they attend to w tokens to the left of the home segment, as well as all tokens within the home segment that is not a future token.

E Details for ImageNet Classification

The CvT Architecture. We implement the CvT model based on a public repository, [4] because this is a concurrent work with no official implementation when we conduct this work. In Table 5, since our CvT re-implementation gets worse test results than reported ones in their arxiv paper, we still list the best test accuracy from Wu et al. [6] for fair comparisons. We report the FLOPs of CvT with our implementation for reasonable comparisons, because our CvT*-LS implementation is based on that. Same as CvT, all the models have three stages where the first stage downsamples the image by a factor of 4 and each of the following stages downsamples the feature map by a factor of 2. CvT*-LS-13 and CvT*-LS-21 have the same configuration as CvT-13 and CvT-21. CvT*-LS-17 and CvT*-LS-21 are our customized models with more layers and higher embedding dimensions in the first two stages ($[3, 4, 10]$, $[3, 4, 14]$ layers respectively and $[128, 256, 768]$ dimensions). We train the model for 300 epochs using a peak learning rate of $5e-4$ with the cosine schedule [65] with 5 epochs of warmup. We use the same set of data augmentations and regularizations as other works including PVT [5] and ViL [11]. In general, CvT*-LS-13 and CvT*-LS-21 closely follow the architectural designs of CvT for fair comparisons. Specifically, in CvT*-LS, we feed the token embeddings extracted by the depth-wise separable convolution [66] of CvT to our long-short term attention. For dynamic projection, we replace W_i^P in Eq. (3) with a depth-wise separable convolution to maintain consistency with the patch embeddings, but we change its BN layer into a weight standardization [67, 68] on the spatial convolution's weights for simplicity. We do not use position encoding. All of our models have 3 stages, and the feature map size is the same as CvT in each stage when the image resolutions are the same. CvT*-LS-13 and CvT*-LS-21 follow the same layer configurations as CvT-13 and CvT-21, i.e., the number of heads, the dimension of each head and the number of Transformer blocks are the same as CvT in each stage. For all models on resolution 224×224, we set $r = [64, 16, 4]$ and $w = [8, 4, 2]$. For higher resolutions, we scale up r and/or w to maintain similar effective receptive fields for the attentions. At resolution 384×384, we use $r = [64, 16, 4]$ and $w = [12, 6, 3]$ for the 3 stages. At resolution 448×448, we use $r = [128, 32, 8]$ and $w = [16, 8, 4]$.

Besides maintaining the CvT architectures, we also try other architectures to further explore the advantage of our method. With the efficient long-short term attention, it becomes affordable to stack more layers on higher-resolution feature maps to fully utilize the expressive power of attention mechanisms. Therefore, we have created two new architectures, CvT*-LS-17 and CvT*-LS-21S, that have more and wider layers in the first two stages, as shown in Table 12. Compared with CvT-21, CvT*-LS-17 has 25% fewer parameters, less FLOPs, but obtained the same level of accuracy. CvT*-LS-21S has fewer parameters than CvT*-LS-21, more FLOPs, and 0.4% higher accuracy, demonstrating the advantage of focusing the computation on higher-resolution feature maps.

The effect of DualLN. We trained the CvT*-LS-13 model without DualLN, which has a test accuracy of 81.3, lower than the 81.9 with DualLN.

[4] https://github.com/rishikksh20/convolution-vision-transformers

Table 12: Architectures of our CvT*-LS-17 and CvT*-LS-21S models. LSTA stands for our Long-Short Term Attention.

	Output Size	Layer Name	CvT*-LS-17	CvT*-LS-21S
Stage 1	56×56	Conv. Embed.	7×7, 128, stride 4	
	56×56	Conv. Proj. LSTA MLP	$\begin{bmatrix} 3 \times 3, 128 \\ H = 2, D = 128 \\ r = 64, w = 8 \\ R = 4 \end{bmatrix} \times 3$	
Stage 2	28×28	Conv. Embed.	3×3, 256, stride 2	
	28×28	Conv. Proj. LSTA MLP	$\begin{bmatrix} 3 \times 3, 256 \\ H = 4, D = 256 \\ r = 16, w = 4 \\ R = 4 \end{bmatrix} \times 4$	
Stage 3	14×14	Conv. Embed.	3×3, 384, stride 2	
	14×14	Conv. Proj. LSTA MLP	$\begin{bmatrix} 3 \times 3, 384 \\ H = 6, D = 384 \\ r = 4, w = 2 \\ R = 4 \end{bmatrix} \times 10$	$\begin{bmatrix} 3 \times 3, 384 \\ H = 6, D = 384 \\ r = 4, w = 2 \\ R = 4 \end{bmatrix} \times 14$

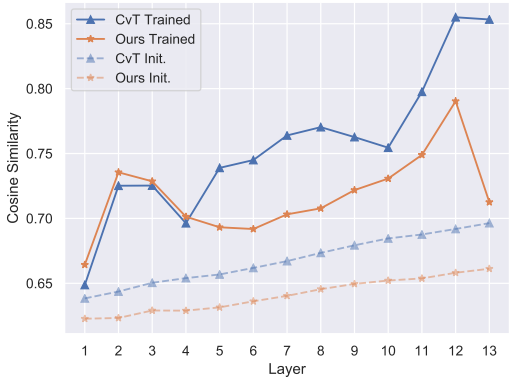

Figure 6: Pairwise cosine similarity between patch embeddings at different layers of CvT-13 and CvT*-LS-13, averaged on 50k images of ImageNet validation set. The larger cosine similarities at deeper layer suggest that the feature representation is less diverse.

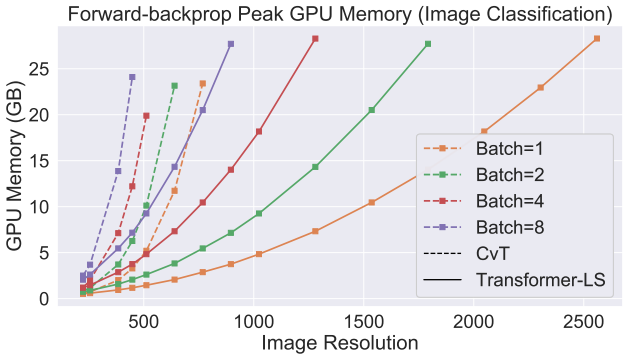

Figure 7: Running memory consumption of full self-attention (CvT-13) and Long-Short Transformer on different tasks. We increase the sequence length resolution until the model is out of memory on a V100 GPU with 32GB memory.

F Evaluate the robustness of models trained on ImageNet-1k.

Table 13: Corruption Error (CE) on ImageNet-C

Arch.	Noise			Blur				Weather				Digital			
	Gauss.	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG
ResNet-50	34.24	49.25	55.84	56.24	57.04	63.53	63.68	64.02	64.04	64.89	69.25	70.72	73.14	75.29	75.76
DeiT-S	26.93	36.81	36.89	39.38	40.14	43.32	43.80	44.36	45.71	46.90	47.27	48.57	52.15	57.53	62.91
CvT*-LS-13	25.64	36.89	37.06	38.06	43.78	43.78	44.62	45.92	47.77	47.91	49.60	49.66	54.92	57.24	68.72
CvT*-LS-17	25.26	35.06	35.48	37.38	41.37	43.95	44.47	46.05	46.17	46.38	49.08	49.37	54.29	54.54	69.54
CvT*-LS-21	24.28	34.95	35.03	35.93	39.86	40.71	41.27	41.78	44.72	45.24	45.50	47.19	51.84	53.78	67.05

Table 14: Robustness evaluation on ImageNet-9. We report Top-1 Accuracy.

Model	Params (M)	ImageNet (%)	ImageNet-9 [59](%)		
			Original	Mixed-same	Mixed-rand
ResNet-50 [35]	25.6	76.2	94.9	87.1	81.6
DeiT-S [36]	22.1	79.8	97.1	89.1	84.2
CvT*-LS-13	20.3	81.9	97.0	90.7	85.6
CvT*-LS-21	32.1	**82.7**	**97.2**	**91.5**	**85.8**

For a fair comparison, we choose models with similar number of parameters. We select two representative models, including the CNN-based model (ResNet) and the transformer-based model (DeiT). We give detailed results on all types of image transforms on ImageNet-C in Table 13. We evaluate our method on various ImageNet robustness benchmarks as follows:

- **ImageNet-C**. ImageNet-C refers to the common corruption dataset. It consists of 15 types of algorithmically common corruptions from noise, blur, weather, and digital categories. Each type contains five levels of severity. In Table 4, we report the normalized mean corruption error (mCE) defined in Hendrycks and Dietterich [56]. In Table 13, we report the corruption error among different types. In both tables, the lower value means higher robustness.
- **ImageNet-A**. ImageNet-A is the natural adversarial example dataset. It contains naturally collected images from online that mislead the ImageNet classifiers. It contains 7,500 adversarially filtered images. We use accuracy as our evaluation metric. The higher accuracy refers to better robustness.
- **ImageNet-R**. ImageNet-R (**R**endition) aims to evaluate the model generalization performance on out-of-distribution data. It contains renditions of 200 ImageNet classes (e.g. cartoons, graffiti, embroidery). We use accuracy as the evaluation metric.
- **ImageNet-9**. ImageNet-9 aims to evaluate the model background robustness. It designs to measure the extent of the model relying on the image background. Following the standard setting [59], we evaluate the two categories, including MIXED-SAME and MIXED-RAND. MIXED-SAME refers to replace the background of the selected image with a random background of the same class by GrabCut [59]; MIXED-RAND refers to replace the image background with a random background of the random class.

From table 6, we find that our method achieves significant improvement compared to CNN-based network (ResNet). For instance, our method improves the accuracy by 23.6%, 22.1%, 9.7% compared to ResNet on ImageNet-C, ImageNet-A, and ImageNet-R, respectively. For ImageNet-9, our method also achieves favorable improvement by 4.3% on average (Mixed-same and Mixed-rand). It indicates that our method is insensitive to background changes. We guess the potential reasons for these improvements are (1) the attention mechanism and (2) the strong data augmentation strategies during the training for vision transformer [4, 36]. The first design helps the model focus more on the global context of the image as each patch could attend to the whole image areas. It reduces the local texture bias of CNN. The latter design increases the diversity of the training data to improve model's generalization ability. Compared to DeiT, we also surprisingly find that our method achieves slightly better performance. One plausible explanation is that our long-term attention has a favorable smoothing effect on the noisy representations. Such improvements also indicate that different designs of attention and network architecture can be essential to improve the robustness. As the goal of this paper is not to design a robust vision transformer, the robustness is an additional bonus of our method.

We believe that our observation opens new directions for designing robust vision Transformers. We leave the in-depth study as an important future work.

The detailed results of ImageNet-C and ImageNet-9 are shown in Table 13 and Table 14 respectively.