LeaPformer: Enabling Linear Transformers for Autoregressive and Simultaneous Tasks via Learned Proportions

Anonymous ACL submission

Abstract

Position-based re-weighting functions have been proposed recently as a promising approach to recover degraded model performance from conventional linearized transformers. However, state-of-the-art re-weighting functions rely heavily on target sequence lengths, making it difficult or impossible to apply to autoregressive and simultaneous tasks where the target and sometimes even the input sequence length are unknown beforehand. To resolve this issue and enable these re-weighting functions for a wider range of tasks, we propose *Learned Proportions* (LeaP) and LeaPformers. Our contribution is built on two major components. First, we generalize the dependence on explicit positional representations and sequence lengths into a dependence on sequence proportions for re-weighting, removing theoretical dependence on sequence lengths. Second, we replace static positional representations with dynamic proportions derived via a compact module, enabling more flexible attention concentration patterns. We validate the potential of LeaPformer against eight representative efficient transformers on the competitive Long-Range Arena benchmark, where LeaPformer achieves the best quality-throughput trade-off. We also demonstrate, for the first time, that position-based re-weighting functions can be applied to simultaneous tasks, achieving competitive results on speech-to-text translation for two language pairs.

1 Introduction

Transformers (Vaswani et al., 2017) became dominant in the natural language processing (NLP) solution space only a few years after inception, demonstrating state-of-the-art performance for a range of applications. With the advent of widely accessible large language models (LLM), transformers as a class of models are being studied more closely than ever. Unfortunately, the quadratic complexity of the attention mechanisms of typical transformers limits the lengths of the sequences that they can process, rendering them sub-optimal or even impossible to apply for tasks with long sequences.

Naturally, an active area of potential improvement for classical transformers are efficient attention mechanisms that reduce the quadratic complexity of typical softmax attention with respect to sequence length. Many efficient transformer variants have been proposed, including truly linear attention mechanisms with no prior environmental assumptions (Katharopoulos et al., 2020; Choromanski et al., 2020; Peng et al., 2021; Chen et al., 2021; Qin et al., 2022). Such mechanisms not only increase throughput for sufficiently long sequences, but also enable larger sequences via the aforementioned throughput increase and, typically, a reduction in a model's peak memory footprint. While the aforementioned linear transformers are typically effective for specific tasks, they tend to exhibit varying degrees of quality degradation when generalized.

To address this issue, re-weighting functions have been recently formalized (Qin et al., 2022) in linear transformers and serve to concentrate attention scores in a manner similar to the softmax operator. Although promising, the state-of-the-art position-based re-weighting functions rely on explicit token positions and sequence length (Su et al., 2022; Qin et al., 2022). This reliance on knowing the sequence length beforehand can make it difficult to apply those re-weighting functions and linear transformers to autoregressive tasks without employing specialized solutions (Agostinelli and Chen, 2023) and renders it impossible to apply them to simultaneous tasks, such as simultaneous translation. Furthermore, existing re-weighting functions' reliance on explicit positional representations usually produce static attention concentration patterns, which can severely limit their general applicability when an attention concentration pattern is ill-suited to a given task.

To solve this reliance on explicit positional representations and enable linear transformers for a wider range of tasks, we propose a novel approach that we refer to as *Learned Proportions* (LeaP) and call models we apply it to LeaPformers. This contribution is composed of two major aspects: proportions and learned behavior. First, we generalize the dependence on explicit positional representations and sequence lengths into a more direct, intuitive dependence on proportions of a sequence for re-weighting. This elaborates on how proportion-based re-weighting functions can be more effective in concentrating attention behavior and removes theoretical dependence on sequence length. Second, instead of employing static positional representations, we construct and deploy a compact module that dynamically derives sequence proportions for a given token during training and inference. These straightforward, but critical, contributions ultimately remove any reliance that current position-based re-weighting functions may have on sequence length, enabling them for tasks where the sequence length is not known beforehand (and cannot be estimated) and/or where attention concentration patterns are more complex.

To validate our proposed approach, we primarily test our LeaPformer against cosFormer, the state-of-the-art position-based linear transformer, by adapting cosFormer's cosine-based re-weighting function via LeaP. We also evaluate and compare with seven other representative attention mechanisms. We apply our LeaPformer to the Long-Range Arena (LRA) benchmark (Tay et al., 2021), a common and competitive benchmark for efficient attention mechanisms on long sequences, and to multiple language pairs for simultaneous speech-to-text translation (SimulST) (Ma et al., 2020b). When compared to popular, previously proposed efficient attention mechanisms on the LRA benchmark, the proposed LeaPformer achieves the best quality-efficiency trade-off, balanced performance across tasks, small memory footprint, and notably beats cosFormer's inference quality. Moreover, when applied to simultaneous translation, LeaPformer demonstrates competitive results with a reasonable quality-efficiency trade-off compared to classical softmax attention for critical ablations, with some variations achieving quality loss of only 0.26 BLEU-4 (Post, 2018) for English to German and 0.23 BLEU-4 for French to English while being completely linear in complexity. To our knowledge, this is the first time that a position-based re-weighting function for linear transformers is successfully applied to simultaneous tasks.

2 Background

2.1 Softmax Attention Mechanisms

As introduced by Vaswani et. al (2017), the multi-headed self-attention in transformers can generally be described by Equations 1 and 2, where query $Q_h = xW_{q,h}$, key $K_h = xW_{k,h}$, and value $V_h = xW_{v,h}$, with $x \in \mathbb{R}^{n \times d_{model}}$ being the input sequence for each attention head that divides the model embedding space d_{model} into some d_{head} (denoted as d hereafter for simplicity) and $W_{q,h} \in \mathbb{R}^{d_{model} \times d}$, $W_{k,h} \in \mathbb{R}^{d_{model} \times d}$ and $W_{v,h} \in \mathbb{R}^{d_{model} \times d}$. In cases where the concatenation of the attention head outputs differs in dimensionality from d_{model}, an optional output projection layer is commonly applied via $W_{out} \in \mathbb{R}^{d_{out} \times d_{model}}$.

$$a_h(x) = softmax(\frac{Q_h K_h^T}{\sqrt{d}})V_h \tag{1}$$

$$A(x) = concat(a_1(x), a_2(x), \ \dots \ , a_H(x))W_{out} \tag{2}$$

For long sequences, the quadratic complexity of the mechanism demonstrated in Equation 1 can prove to be a throughput bottleneck both during training and inference.

2.2 Efficient and Linear Transformers

Efficient and/or linear transformers have emerged over the past few years as an active area of research for particularly resource or latency-constrained environments, exhibiting notable inference speedups and smaller memory footprints. These transformer variants focus on alternative attention mechanisms that reduce the quadratic complexity of typical softmax attention. A plethora of efficient transformer options exist that can be classified into a few groups: sliding-window or localized attention mechanisms (Dai et al., 2019; Parmar et al., 2018; Wu et al., 2020; Beltagy et al., 2020), pattern or sparsity-based attention mechanisms (Child et al., 2019; Zaheer et al., 2020), kernel-based and truly linear attention mechanisms with no priors (Katharopoulos et al., 2020; Choromanski et al., 2020; Qin et al., 2022; Peng et al., 2021; Chen et al., 2021), and some unique outliers (Wang et al., 2020c; Kitaev et al., 2020).

Truly linear transformers with no prior environmental assumptions (i.e. no assumed sparsity, local dependencies, etc.) are typically kernel-based substitutions for the softmax mechanism. This can be described via row-wise outputs for each attention head in Equations 3 to 5, with S corresponding to any similarity function that transforms the product of the query and key matrices. When S is equal to exp, Equation 3 is an accurate representation of softmax attention. Alternatively, when S is decomposable via a S_q and S_k, as seen in Equation 4, computation can be reordered such that the attention complexity changes from $O(N_1N_2d)$ in Equation 4 when multiplying QK^T first, to $O(N_1d^2+N_2d^2)$ to in Equation 5 when multiplying K^TV first. Here, N_1 corresponds to the length of the query matrix and N_2 corresponds to the lengths of the key and value matrices. When N_1 or N_2 are significantly larger than d, this rearrangement of the attention calculation leads to linear complexity with respect to the sequence length.

$$a_{h,i}(x) = \sum_j \frac{exp(Q_iK_j^T)}{\sum_j exp(Q_iK_j^T)}V_j \tag{3}$$

$$\begin{gathered}\tilde{a}_{h,i}(x) = \sum_j \frac{S(Q_iK_j^T)}{\sum_j S(Q_iK_j^T)}V_j \\ S(Q_iK_j^T) = S_q(Q_i)S_k(K_j^T)\end{gathered} \tag{4}$$

$$\tilde{a}_{h,i}(x) = \sum_j \frac{S_q(Q_i)(S_k(K_j^T)V_j)}{S_q(Q_i)\sum_j S_k(K_j^T)} \tag{5}$$

2.3 Position-Based Re-weighting Functions for Linear Transformers

While achieving linearity, the aforementioned works usually suffer from varying degrees of degraded model performance. To address this, re-weighting functions have been recently proposed that introduce an additional function to augment $S(Q_i, K_j^T)$, with the express purpose of concentrating/adjusting the probability distribution of the normalized QK^T (Qin et al., 2022). Re-weighting functions are commonly based on token positions and can be applied via Equation 6 as $\sigma(i,j)$:

$$S(Q_i, K_j^T) = S_q(Q_i)S_k(K_j^T)\sigma(i,j) \tag{6}$$

Note that even though $\sigma(i,j)$ in Equation 6 is placed at the end of the sequence, that placement is arbitrary. For example, placing $\sigma(i,j)$ in between or before the transformed query and key matrices would also be valid as a re-weighting addition. $\sigma(i,j)$ can also map to any number of possible concentration methods, such as a matrix or scalar value modifying $S(Q_i, K_j^T)$.

Elaborate position-based encoding schemes (Raffel et al., 2020; Wang et al., 2020a; Wang and Chen, 2020; Liutkus et al., 2021), using absolute or relative token positions, have advanced the scheme utilized by original transformers (Vaswani et al., 2017) and many provide what can be intuited as position-based re-weighting functions. However, those schemes are specifically designed for the $S(Q_i, K_j^T)$ formulation and do not work for the decomposed $S_q(Q_i)S_k(K_j^T)$ linearized formulation.

Rotary Positional Embeddings (RoPE) (Su et al., 2022), with some minor modifications, is closest to being a true position-based re-weighting function for linear transformers by using absolute token positions. As demonstrated in Equation 7, R represents RoPE's re-weighting function acting as a rotational transform and θ represents the set of rotation constants defined by head dimensionality d.

$$\begin{gathered}\sigma(i,j) = R^d_{\theta,j-i} = (R^d_{\theta,i})^T R^d_{\theta,j} \\ S(Q_i, K_j^T) = S_q(Q_i)\sigma(i,j)S_k(K_j^T) \\ S(Q_i, K_j^T) = (S_q(Q_i)(R^d_{\theta,i})^T)(R^d_{\theta,j}S_k(K_j^T))\end{gathered} \tag{7}$$

Notably, RoPE is unaware of the total sequence length when it is applied, which can cause intuitive qualitative problems. For example, RoPE would treat two tokens that are 100 tokens apart in a 1k length sequence and a 200 length sequence the same, where the actual relationship of the two tokens could change drastically in the two sequences. In practice, RoPE's re-weighting function still works reasonably well, as many sequences exhibit locality (closer tokens are more related) which can be captured by $\sigma(i,j)$. Nevertheless, the lack of sequence length makes RoPE's re-weighting ability inherently limited, especially for sequences that exhibit more than the locality characteristic.

Recently introduced, cosFormer (Qin et al., 2022) is the state-of-the-art in position-based linear transformers that utilizes sequence length in addition to absolute token position. cosFormer's proposed mechanism, with common-sense modifications (Agostinelli and Chen, 2023), is described by Equation 8. Here, S_q and S_k are set to $ReLU$ and their cosine-based re-weighting function is distributable via Ptolemy's method for expanding trigonometric expressions. Intuitively, when the positions i and j of two tokens are closer, the cosine's response is increased, emphasizing locality. Conversely, when the two positions are far apart, the response approaches zero, representing maximum attenuation via re-weighting. In spite of its inherent locality bias, cosFormer has been empirically shown to form longer-range dependencies than local attention implementations might allow.

$$S(Q_i, K_j^T) = S_q(Q_i)S_k(K_j^T)cos(\frac{\pi}{2}(\frac{i}{N_1} - \frac{j}{N_2})) \tag{8}$$

Unlike RoPE, cosFormer can recognize differences in token distances relative to the sequence length, re-weighting more dynamically in practice. Using our previous example, cosFormer would treat two tokens that are 100 positions apart differently in a 1k length sequence versus a 200 length sequence, an intuitive improvement.

2.4 Motivation for Further Investigation

Unfortunately, the reliance on sequence length causes critical problems for autoregressive and simultaneous tasks. For instance, it can be difficult to apply re-weighting functions to autoregressive tasks where target sequence lengths are usually not known beforehand (e.g. translation, TTS). Although some effort has been made to address these issues (Agostinelli and Chen, 2023; Liu et al., 2022), mostly via target sequence length prediction based on the full input sequence, proposed solutions are prone to some level of approximation error. Furthermore, none of the proposed solutions has addressed the impossibility of being applied to simultaneous tasks, where even the full input sequence is not available at decoding time-steps.

Moreover, the static nature of the state-of-the-art's re-weighting functionality can cause issues from an inference quality standpoint. cosFormer's re-weighting function focuses on encouraging locality, but this can be problematic when locality bias is not important to a given application. RoPE and several similar schemes suffer from the same problem. In such instances, more dynamic flexibility in the re-weighting function to encourage strong, long-range connections would be preferred. An example of when this flexibility may be desirable can be found in a typical translation task, when encoding or decoding in languages like German that tend to exhibit subject-object-verb (SOV) structures as opposed to subject-verb-object (SVO) structures in languages like English and may require diverse attention patterns and long-range dependencies. A verb near the end of a German sentence may attend strongly to the subject near the beginning of the sentence, but static re-weighting functions like the one employed by cosFormer would likely have trouble enabling this relationship.

3 LeaPformer: Learned Proportions for Linear Transformer Re-weighting

To address issues with the state-of-the-art re-weighting functions for linear transformers for both autoregressive and simultaneous tasks, we provide two concrete contributions. First, we generalize the reliance on absolute token position and sequence length into a more direct, intuitive reliance on the relative placement of a token in the sequence which we refer to as a *proportion*. This generalization allows for easier analysis of re-weighting function behavior and removes theoretical dependence on sequence length. Additionally, we propose, construct, and deploy a compact module to learn proportional representations derived from each token, a technique that we call *Learned Proportions* (LeaP) and call the models it is applied to LeaPformers. LeaPformers can be applied to tasks where sequence lengths are unknown and, more importantly, are capable of capturing dynamic attention patterns over static position-based re-weighting functions.

3.1 From Position and Sequence Length to Proportion

We define proportion-based re-weighting in Equation 9, where P_q and P_k represent proportions of sequences from which queries and keys are derived from and $\sigma(P_{q,i}, P_{k,j})$ represents the re-weighting function with a reliance on the provided proportions. Technically, P_q and P_k can be set in any manner, but for the most straightforward proportion-based re-weighting implementations, they would correspond to the proportion of a sequence that a token is placed (e.g., at 20% of the sequence).

$$\begin{gathered} P_q = [P_{q,1},\ P_{q,2},\ \dots\ P_{q,N_1}], \quad 0 \leq\ P_{q,i} \leq 1 \\ P_k = [P_{k,1},\ P_{k,2},\ \dots\ P_{k,N_2}], \quad 0 \leq P_{k,j} \leq 1 \\ S(Q_i, K_j^T) = S_q(Q_i)S_k(K_j^T)\sigma(P_{q,i}, P_{k,j}) \end{gathered} \tag{9}$$

Under this definition, cosFormer's formulation in Equation 8 can be considered as a special case, where we replace $\sigma(P_{q,i}, P_{k,j})$ in Equation 9 with the cosine-based re-weighting function of cosFormer and define P_q and P_k as being explicit token positions divided by the sequence length, as shown in Equation 10:

$$P_q = [\frac{1}{N_1}, \ \dots \ , 1], \quad P_k = [\frac{1}{N_2}, \ \dots \ , 1]$$
$$S(Q_i, K_j^T) = S_q(Q_i)S_k(K_j^T)cos(\frac{\pi}{2}(P_{q,i} - P_{k,j})) \tag{10}$$

3.2 LeaP and LeaPformer: Learned Proportions

In addition to determining the proportions statically as in the case of cosFormer in Equation 10, we propose that models can actually learn to derive these representative proportions via a module containing a compact network embedded within attention blocks. We call this method *Learned Proportions* (LeaP) and models utilizing this technique LeaPformers. The possible inference quality benefits of LeaP can be understood intuitively. Suppose that P_k is set in a static manner in accordance with explicit positional representations, but P_q is derived via a small module based on the query matrix. The module's learned behavior could produce derived elements of P_q equal to classical positional representations, thus replicating the behavior and performance of attention mechanisms like cosFormer, but could alternatively defer the inter-token relationships that cosFormer might otherwise emphasize (i.e. an emphasis on locality). Along these lines, we can redefine the aforementioned proportions in accordance with Equation 11, where $LeaP_Q$ and $LeaP_K$ represent the proposed modules that derive proportions based on the query and key matrices, and P_q and P_k are redefined as $P_q(Q)$ and $P_k(K)$.

$$\begin{aligned} P_q(Q_i) &= P_{q,i} = LeaP_Q(Q_i) \\ P_k(K_j) &= P_{k,j} = LeaP_K(K_j) \\ P_q(Q) &= [LeaP_Q(Q_1), \ \dots \ , LeaP_Q(Q_{N_1})] \\ P_k(K) &= [LeaP_K(K_1), \ \dots \ , LeaP_K(K_{N_2})] \end{aligned} \tag{11}$$

To demonstrate potential inference quality benefits further, we can refer back to our example of translation to or from German and the SOV structure that cosFormer would likely struggle to model well. If P_q is derived from a small LeaP module in self-attention, models could effectively defer the locality bias inherent to cosFormer to elsewhere in the sequence. If correctly learned, this might allow models to defer their attention concentration from the verb at the end of the German sequence to the beginning of the sequence, where we might expect a typically strong attention score. Allowing derivations of both P_q and P_k would, naturally, afford maximum flexibility in attention patterns produced by the employed re-weighting function.

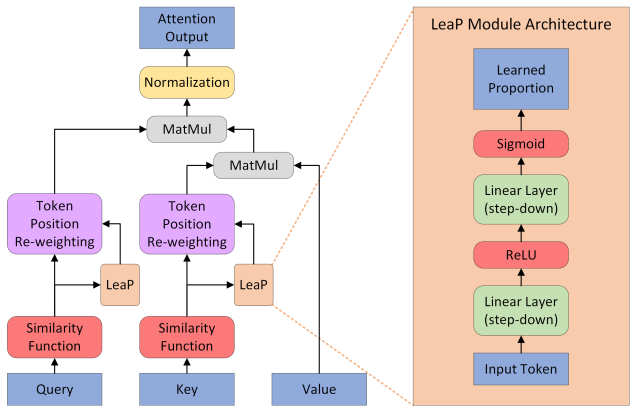

Figure 1: Illustration of the proposed *Learned Proportions* (LeaP) augmentation to linear transformer attention mechanisms. The LeaP module takes each token of the query and key matrices and steps down their dimensionality across two linear layers to a single proportion.

Beyond the possible benefits of LeaP in terms of inference quality, this method removes any dependence that proportion-based re-weighting functions have on knowing the sequence length beforehand, widely enabling them for autoregressive tasks without target sequence length prediction and, for the first time, rendering it possible to apply them to simultaneous tasks.

3.3 Proposed LeaP Module Architecture

It is critical that the addition of LeaP does not significantly affect the throughput of a given model or its memory footprint, as it is intended for resource-constrained and latency-sensitive environments. Given that, we recommend a module composed of a simple, two-layer feed-forward network that steps down the attention head embedding dimension with a ReLU activation between the layers and a sigmoid activation at the end of the network, along the lines of the augmentation highlighted in Figure 1. The choice of a ReLU activation is based on empirical tests on the Long-Range Arena benchmark (Tay et al., 2021) which determined that, as opposed to several other competitive options, ReLU generalized well to multiple tasks.

While a separate LeaP module for each attention head would be natural, we found in our tests that this made a very minor difference in terms

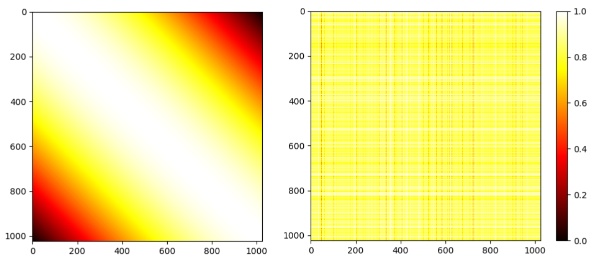

Figure 2: An example of re-weighting matrices across all query (y-axis) and key (x-axis) token positions for baseline cosFormer (left) and LeaPformer (right) on list-operations in the Long-Range Arena benchmark. In this example, LeaPformer has clearly learned to attenuate more dynamically.

of quality. Indeed, even for English to German SimulST, we observed that when replacing the decoder self-attention block with LeaPformer where a separate LeaP module was provided for each attention head the models were roughly equivalent in quality. Given that, we share one module for all heads.

3.4 Observed LeaP Module Expressivity

Given the activation functions chosen for the LeaP module's architecture, it is important to examine the expressivity of the module, ensuring that it outputs a complex range of values as opposed to saturating to values of 0 or 1. We provide an example in Figure 2 that compares the re-weighting matrices of baseline cosFormer and LeaPformer for the list-operations task in the LRA benchmark, a fairly difficult one. As can be observed in the example, and as we generally found in practice, the baseline cosFormer can only provide static re-weighting emphasizing locality (with the largest weights along diagonal for the same position); in contrast, cosFormer augmented with LeaP is capable of generating complex re-weighting matrices that lightly attenuate between most positions while selectively attenuating harshly or not at all. The fact that there is wide-spread, light attenuation across several examples indicates that the module is very capable of avoiding saturation.

4 Experiments

We validate the potential of LeaP by applying it to cosFormer on two major sets of tasks, and all references to LeaPformers in the following sections refer to this augmentation of cosFormers. The two sets of tasks include the popular Long-Range Arena (LRA) benchmark (Tay et al., 2021), built specifically for validating the capabilities of efficient attention mechanisms. We also validate LeaPformers on speech-to-text simultaneous translation (SimulST) via wait-k (Ma et al., 2019, 2020a,b) across two language pairs. For our SimulST experiments, we employ Fairseq (Ott et al., 2019) for training and validation alongside SimulEval (Ma et al., 2020c) for evaluation. LRA results are compared via accuracy and SimulST results are compared via detokenized BLEU-4 (called BLEU later) using sacreBLEU (Post, 2018). Additional details can be found in the Appendix.

4.1 Long-Range Arena Benchmark Setup

To better compare evaluated models, on top of the Long-Range Arena (LRA) benchmark provided by (Tay et al., 2021), our implementation follows Skyformer's (Chen et al., 2021) PyTorch framework and reuses their architectures and hyperparameters, which we hold static. We provide baseline results for a number of architectures, including the classical transformer (Vaswani et al., 2017) alongside several efficient transformers.

Regarding the LeaPformers tested on the LRA benchmark, a minimal setup was initially employed with around a maximum of a 0.2% increase on the number of parameters for the LeaP module. Additionally, a larger module was employed with a maximum increase of 1.5% to the number of parameters to investigate the effects of increased size. Some very limited fine-tuning was employed across a few possible module sizes on a per-task basis for the larger LeaPformer, depending on the perceived difficulty of the task.

4.2 Long-Range Arena Benchmark Results

A holistic view of overall performance is well-described by Figure 3, with kernel-based linear transformers tending to provide an excellent quality-throughput trade-off. As clearly demonstrated in the figure, LeaPformer provides the best trade-off, exhibiting significant quality increases over Linear Transformer and overall supremacy compared to Performer, Linformer, Reformer, and Skyformer, with a minimal memory footprint. Details on inference quality are showcased in Table 1, where both LeaPformer-0.2% and LeaPformer-1.5% exhibit a balanced performance profile. While classical softmax attention achieves the highest average score by a notable margin, it is beaten on a number of tasks by other methods.

Compared to cosFormer, LeaPformer provides,

Attention Mechanism	ListOps	Text Cls.	Text Rtr.	Path-32	Img. Cls.	Avg.
Softmax Attn. (Vaswani et al., 2017)	37.94	60.51	80.52	**75.54**	<u>41.74</u>	**59.25**
Linear Attn. (Katharopoulos et al., 2020)	**39.21**	61.53	78.78	68.23	39.14	57.38
Linformer (Wang et al., 2020c)	37.04	57.65	77.61	57.91	37.85	53.61
Performer (Choromanski et al., 2020)	38.17	64.24	80.11	68.54	37.42	57.70
BigBird (Zaheer et al., 2020)	38.36	60.72	<u>80.97</u>	<u>72.80</u>	40.37	58.64
Reformer (Kitaev et al., 2020)	36.44	63.14	78.63	69.29	**42.85**	58.07
Skyformer (Chen et al., 2021)	38.66	**65.38**	**81.77**	68.74	36.07	58.12
RoPE w/ Linear Attn. (Su et al., 2022)	38.31	64.79	77.54	67.61	39.17	57.48
cosFormer (Qin et al., 2022)	<u>38.96</u>	61.66	79.29	68.96	38.26	57.43
LeaPformer-0.2%	38.26	64.70	79.88	70.76	38.26	58.37
LeaPformer-1.5%	<u>38.96</u>	<u>64.90</u>	80.62	68.99	40.00	<u>58.69</u>

Table 1: Quality results on the Long-Range Arena benchmark. All results are measures of accuracy (higher is better) and are weighted evenly for the purpose of the average score. Best results are **bolded**, second best results are <u>underlined</u>. Both LeaPformer variants showcase competitive performance across a range of tasks, with LeaPformer-1.5% achieving the second best average score, beating all other non-quadratic transformers.

Attention Mechanism	Training Thrpt. (itr/sec)		
	1K	2K	4K
Softmax Attn.	14.08	6.03	1.64
Linear Attn.	68.00	28.43	15.18
Linformer	48.96	20.49	11.36
Performer	38.83	17.72	9.02
BigBird	15.97	6.76	3.52
Reformer	32.07	15.08	7.71
Skyformer	26.02	12.36	6.06
RoPE w/ Linear Attn.	48.90	23.81	12.27
cosFormer	58.91	25.64	13.13
LeaPformer-0.2%	56.30	24.72	12.81
LeaPformer-1.5%	53.58	23.39	11.76

Table 2: Efficiency results on the Long-Range Arena benchmark. Training throughput values (higher is better, inference speed is identical) are provided for various sequence lengths defined by the five tasks in the benchmark.

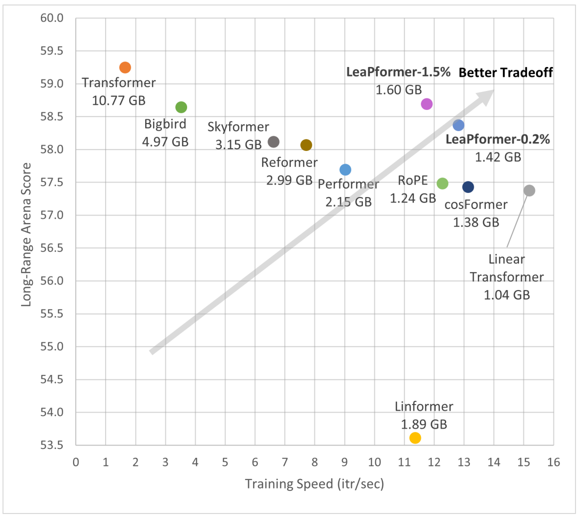

Figure 3: Performance of various linear attention mechanisms on the Long-Range Arena benchmark. Throughput for 4k sequence length tasks (x-axis) is set against average score (y-axis) across the five tasks in the benchmark with a provided maximum memory footprint.

at a minor throughput and negligible memory footprint penalty, significant increases to scores across some of cosFormer's most problematic tasks, including large improvements for text and image classification. Additionally, when compared to the score profiles of other efficient attention mechanisms, LeaPformer does not seem to specialize nearly as much as other architectures (aside from some difficulty on the pathfinding task), indicating its balanced performance. In terms of inference quality, BigBird is the closest to providing a similarly balanced performance profile, but this comes with significant throughput reductions as shown in Table 2 and noticeable increases to memory footprint. LeaPformer matches the general inference quality of task-balanced models with a massively reduced memory footprint while still exhibiting a minimum 1.52x throughput increase over those mechanisms.

4.3 SimulST Setup

For the purposes of our SimulST related experiments, we employed a model inspired by the ESPnet-ST toolkit (Inaguma et al., 2020) that focused on end-to-end speech-to-text (S2T) translation with a modified cross-attention block for a wait-k and fixed pre-decision paradigm (Ma et al.,

Attention Mechanism	ppl(tr)	ppl(dev)
cosFormer Dec. Self-Attn.	8.44	9.86
LeaPformer Dec. Self-Attn.	7.86	9.40

Table 3: Brief comparison of cosFormer and LeaPformer trained on MuST-C en-de. Perplexity (lower is better) is generated on the training and validation sets. BLEU scores are not provided for baseline cosFormer as it is impossible to apply to simultaneous tasks (i.e. BLEU scores of near zero) without augmentations.

Attention Mechanism	BLEU	ppl(dev)
Softmax Attention	15.07	9.36
LeaPformer Enc. Self-Attn.	12.00	11.50
LeaPformer Dec. Self-Attn.	14.81	9.40
LeaPformer Cross-Attn.	13.95	11.02
LeaPformer All Attn.	11.19	14.67
ReLU Enc. Self-Attn.	11.55	11.98
ReLU Dec. Self-Attn.	14.67	9.55
ReLU Dec. Cross-Attn.	13.84	11.24
ReLU All Attn.	10.38	15.48

Table 4: Results from SimulST for MuST-C en-de for various LeaPformer and simple ReLU ablations with softmax as a baseline. BLEU scores (higher is better) are generated on the tst-COMMON split.

Attention Mechanism	BLEU	ppl(dev)
Softmax Attention	14.51	9.99
LeaPformer Enc. Self-Attn.	11.18	12.50
LeaPformer Dec. Self-Attn.	14.28	10.11
LeaPformer Cross-Attn.	13.25	11.64
LeaPformer All Attn.	9.69	16.28

Table 5: Results from SimulST for CoVoST fr-en for various LeaPformer ablations with softmax as a baseline. BLEU scores are generated on the recommended, but shortened, test split.

2019, 2020a,b). All model encoders were pre-trained on automatic speech-recognition (ASR) and were trained on a wait-k of 5 and a fixed predecision ratio of 9 and were evaluated on a wait-k of 3 (a slightly larger k for training is suggested by several prior works) with greedy decoding. Models are evaluated via validation set perplexity and by detokenized BLEU-4 (Post, 2018) via SimulEval (Ma et al., 2020c). Two language pairs and two datasets were employed to test the application of LeaPformer to simultaneous tasks. We utilized MuST-C's (Cattoni et al., 2021) English to German (en-de) split and CoVoST 2's (Wang et al., 2020b) French to English (fr-en) split. More comprehensive evaluation is provided for the en-de pair, comparing the results of LeaPformer to an ablation without a re-weighting function.

4.4 SimulST Results

We first seek to showcase quality gains from LeaPformer when compared to baseline cosFormer, justifying its inclusion not only from the perspective of necessity but also as an overall improvement. Table 3 demonstrates the results of a brief comparison on en-de simultaneous translation (note that cosFormer can still be employed for training, where sequence lengths are known), where significant quality improvement is observed. Having established the capability of the proposed method, we seek to validate it further on en-de simultaneous translation while also providing several ablations for LeaPformer, representing a wide-range of quality-throughput trade-offs. Additionally, we seek to show that applying the LeaP-augmented re-weighting function of LeaPformer is consistently useful by testing models trained without any re-weighting functionality, operating as a variation on Linear Transformer (Katharopoulos et al., 2020). Table 4 showcases the results of this study, where LeaPformer ablations consistently beat their simple ReLU-based alternative. The most competitive ablation in terms of translation quality emerges as a model with the decoder self-attention block replaced by LeaPformer, achieving only a 0.26 BLEU reduction compared to softmax attention.

Similar results are provided for the fr-en language pair in Table 5, with trends from en-de generally persisting. The most competitive translation quality ablations continue to be replacements of the decoder self-attention blocks with LeaPformer, where only a 0.23 BLEU reduction was observed.

5 Conclusion

In this paper, we made two concrete contributions. We re-framed dependencies on explicit positional representations and sequence lengths to dependencies on sequence proportions, removing theoretical dependence on sequence lengths. Additionally, we proposed LeaPformers and applied them to the state-of-the-art in proportion-based linear transformers, cosFormer, achieving the best performance trade-off on the Long-Range Arena benchmark. Moreover, we applied proportion-based transformers for the first time to simultaneous translation, achieving minimal quality loss compared to softmax attention for two language pairs.

6 Limitations

Regarding the limitations of this work, there are a few areas that remain less explored and could be elaborated upon. A direction for future work might focus on a more exhaustive examination of possible LeaP module architectures. While the proposed structure works well in practice, the design is simple and could likely be improved upon. Additionally, a wider breadth of applications could be explored to validate LeaPformers. While nothing precludes our proposed LeaP module and LeaPformers from being applied elsewhere, we focused on validating it on the competitive Long-Range Arena benchmark and simultaneous speech-to-text translation, especially given the associated novelty of the latter task and our method's application to it.

References

Victor Agostinelli and Lizhong Chen. 2023. Improving autoregressive nlp tasks via modular linearized attention.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer.

Roldano Cattoni, Mattia Antonino Di Gangi, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2021. Must-c: A multilingual corpus for end-to-end speech translation. *Computer Speech & Language*, 66:101155.

Yifan Chen, Qi Zeng, Heng Ji, and Yun Yang. 2021. Skyformer: Remodel self-attention with gaussian kernel and nyström method.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2020. Rethinking attention with performers.

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

Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Enrique Yalta Soplin, Tomoki Hayashi, and Shinji Watanabe. 2020. Espnet-st: All-in-one speech translation toolkit.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer.

Zexiang Liu, Dong Li, Kaiyue Lu, Zhen Qin, Weixuan Sun, Jiacheng Xu, and Yiran Zhong. 2022. Neural architecture search on efficient transformers and beyond.

Antoine Liutkus, Ondřej Cífka, Shih-Lun Wu, Umut Şimşekli, Yi-Hsuan Yang, and Gaël Richard. 2021. Relative positional encoding for transformers with linear complexity.

M. Ma, L. Huang, H. Xiong, R. Zheng, K. Liu, B. Zheng, C. Zhang, Z. He, H. Liu, X. Li, H. Wu, and H. Wang. 2019. Stacl: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics (ACL).

X. Ma, J. Pino, J. Cross, L. Puzon, and J. Gu. 2020a. Monotonic multihea attention. In *International Conference on Learning Representations*.

X. Ma, J. Pino, J. Cross, L. Puzon, and J. Gu. 2020b. Simulmt to simulst: Adapting simultaneous text translation to end-to-end simultaneous speech translation. In *Proceedings of 2020 Asia-Pacific Chapter of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*.

Xutai Ma, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. 2020c. Simuleval: An evaluation toolkit for simultaneous translation.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer.

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

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. 2022. cosformer: Rethinking softmax in attention. In *International Conference on Learning Representations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer.

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

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2022. Roformer: Enhanced transformer with rotary position embedding.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Benyou Wang, Donghao Zhao, Christina Lioma, Qiuchi Li, Peng Zhang, and Jakob Grue Simonsen. 2020a. Encoding word order in complex embeddings.

Changhan Wang, Anne Wu, and Juan Pino. 2020b. Covost 2: A massively multilingual speech-to-text translation corpus.

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

Yu-An Wang and Yun-Nung Chen. 2020. What do position embeddings learn? an empirical study of pre-trained language model positional encoding.

Qingyang Wu, Zhenzhong Lan, Kun Qian, Jing Gu, Alborz Geramifard, and Zhou Yu. 2020. Memformer: A memory-augmented transformer for sequence modeling.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences.

A Appendix

A.1 Licensing Information

Fairseq (Ott et al., 2019) is MIT-licensed and widely available for non-commercial use.

A.2 Codebase and Artifacts

The codebase used for our experiments and proposed method will be released upon publication.

A.3 Hardware Details for Training and Evaluation

All models were trained and evaluated on two NVIDIA Tesla V100 GPUs, except for during evaluation via SimulEval where they operated on a Intel Xeon Platinum 8168 CPU.

A.4 Computational Costs of Experimentation

We estimate that results related to the LRA benchmark required approximately 30 GPU hours to gather with perhaps another 60 GPU hours related to experimentation. Concerning SimulST, we estimate that approximately 18 GPU days were required to generate the results with another 4 GPU days related to experimentation. The aforementioned values are normalized for single GPU-usage.

A.5 RoPE with Linear Attention Elaboration

While not fully elaborated upon in the paper, we provide data for a single possible RoPE (Su et al., 2021) linear transformer by augmenting the seminal Linear Transformer (Katharopoulos et al., 2020) with rotary positional embedding. The provided results for this model are based on one with no additional adaptations towards linear transformer functionality beyond what is mentioned in the original publication detailing RoPE. In our tests, additional assurances (e.g. summation to unity in rows of attention matrix, were it to be calculated) did not significantly affect results.

A.6 Model Architectures and Hyperparameters

Below, we list all architectural details and relevant training hyperparameters to reproduce our experiments. Aside from models explicitly including RoPE in our tests, all other models employed absolute positional encoding (APE).

A.6.1 LRA: ListOps

Below are the architectural details for our ListOps models on the LRA benchmark:

- Encoder Layers: 2
- Transformer Dim. d_{model}: 64
- Attention Heads: 2
- FFN Hidden Dim. d_{ffn}: 128
- LeaP Downsample Factor: 1

The models for LRA ListOps, were optimized with Adam with classical parameters. The models were trained with batches of size 32, warmed up for 1000 updates and linearly climbing to a learning rate of 1e-4. A linear learning rate decay was employed with 20000 updates in total. A CLS token was used for classification. Dropouts of 0.1 were employed when applicable.

A.6.2 LRA: Pathfinder-32

Below are the architectural details for our Pathfinder-32 models on the LRA benchmark:

- Encoder Layers: 2
- Transformer Dim. d_{model}: 64
- Attention Heads: 2
- FFN Hidden Dim. d_{ffn}: 128
- LeaP Downsample Factor: 1

The models for LRA Pathfinder-32, were optimized with Adam with classical parameters. The models were trained with batches of size 128, warmed up for 300 updates and linearly climbing to a learning rate of 2e-4. A linear learning rate decay was employed with 50000 updates in total. A CLS token was used for classification. Dropouts of 0.1 were employed when applicable.

A.6.3 LRA: Text Retrieval

Below are the architectural details for our Text Retrieval models on the LRA benchmark:

- Encoder Layers: 2
- Transformer Dim. d_{model}: 64
- Attention Heads: 2
- FFN Hidden Dim. d_{ffn}: 128
- LeaP Downsample Factor: 2

The models for LRA Text Retrieval, were optimized with Adam with classical parameters. The models were trained with batches of size 16, warmed up for 800 updates and linearly climbing to a learning rate of 2e-4. A linear learning rate decay was employed with 50000 updates in total. A CLS token was used for classification. Dropouts of 0.1 were employed when applicable.

A.6.4 LRA: Text Classification

Below are the architectural details for our Text Classificaiton models on the LRA benchmark:

- Encoder Layers: 2
- Transformer Dim. d_{model}: 64
- Attention Heads: 2
- FFN Hidden Dim. d_{ffn}: 128
- LeaP Downsample Factor: 2

The models for LRA Text Classification, were optimized with Adam with classical parameters. The models were trained with batches of size 32, warmed up for 100 updates and linearly climbing to a learning rate of 2e-4. A linear learning rate decay was employed with 20000 updates in total. A CLS token was used for classification. Dropouts of 0.1 were employed when applicable.

A.6.5 LRA: Image Classification

Below are the architectural details for our Image Retrieval models on the LRA benchmark:

- Encoder Layers: 2
- Transformer Dim. d_{model}: 64
- Attention Heads: 2
- FFN Hidden Dim. d_{ffn}: 128
- LeaP Downsample Factor: 1

The models for LRA Image Retrieval, were optimized with Adam with classical parameters. The models were trained with batches of size 256, warmed up for 200 updates and linearly climbing to a learning rate of 1e-4. A linear learning rate decay was employed with 30000 updates in total. A CLS token was used for classification. Dropouts of 0.1 were employed when applicable.

A.6.6 SimulST Models

Below are the architectural details for our SimulST models:

- Encoder Layers: 12
- Decoder Layers: 6
- Transformer Dim. d_{model}: 256
- Attention Heads: 8
- FFN Hidden Dim. d_{ffn}: 1024
- Conv. Pre-net Layers: 2
- Conv. Pre-net Kernel Size: 3
- Conv. Pre-net Stride: 2
- LeaP Downsample Factor: 4

The models for SimulST tasks were optimized via Adam with classical parameters and a learning rate set to 6e-4 with an identical learning rate scheduler. The models were trained with dynamic batching, warmed up for 6000 updates, starting with a learning rate of 1e-4, and trained for around 18000 updates in total with gradients clipped to 10.0. Dropouts of 0.1 were used for all linear layers and attention. SimulST models were trained with a wait-k of 5 and pre-decision ratio of 9.