Linear Complexity Randomized Self-attention Mechanism

Lin Zheng [1 *] Chong Wang [2] Lingpeng Kong [1 3]

Abstract

Recently, random feature attentions (RFAs) are proposed to approximate the softmax attention in linear time and space complexity by linearizing the exponential kernel. In this paper, we first propose a novel perspective to understand the bias in such approximation by recasting RFAs as self-normalized importance samplers. This perspective further sheds light on an *unbiased* estimator for the whole softmax attention, called randomized attention (RA). RA constructs positive random features via query-specific distributions and enjoys greatly improved approximation fidelity, albeit exhibiting quadratic complexity. By combining the expressiveness in RA and the efficiency in RFA, we develop a novel linear complexity self-attention mechanism called linear randomized attention (LARA). Extensive experiments across various domains demonstrate that RA and LARA significantly improve the performance of RFAs by a substantial margin.

1. Introduction

Transformers (Vaswani et al., 2017) are powerful neural networks for sequence modeling. They have been successfully applied in various domains, such as natural language processing (Vaswani et al., 2017; Dehghani et al., 2019; Devlin et al., 2019; Raffel et al., 2020), computer vision (Carion et al., 2020; Dosovitskiy et al., 2021; Liu et al., 2021), bioinformatics (Rives et al., 2021; Jumper et al., 2021) and reinforcement learning (Chen et al., 2021c). The core building block of transformer models is the self-attention mechanism, which captures complex interactions among sequence elements (Vaswani et al., 2017).

However, the computational complexity of attention mechanism is quadratic in the number of tokens, making it prohibitive to process long sequences. In the past two years, there has been a community effort towards developing efficient attention architectures with improved computation complexity and memory usage (Tay et al., 2020b). Among them, one prominent is to view the attention mechanism through kernelization (Katharopoulos et al., 2020; Choromanski et al., 2021; Peng et al., 2021b, *inter alia*). In this work, we focus on random feature attentions (RFAs) (Peng et al., 2021b; Choromanski et al., 2021), which approximate softmax attention by linearizing the exponential kernel into a dot product of random feature maps. Despite achieving linear time and space complexity, this approximation is biased to the softmax attention as a whole.[1]

In this work, we revisit RFA and show that it can be reinterpreted as a self-normalized importance sampler to softmax attention. This insight reveals that the source of the approximation bias in RFAs comes from the self-normalization in estimation (Owen, 2013). We further show softmax attention can be written as an expectation of linearized attention over an input-dependent mixture distribution. These findings suggest that we can in principle construct an unbiased estimator for the softmax attention as a whole, as opposed to merely exponential kernels in previous work. We call such unbiased estimation *randomized attention* or RA. To the best of our knowledge, this is the first unbiased approximation of the whole softmax attention via kernel linearization.

RA constructs positive random features via distributions exclusive to each query. Since RFAs only employ an input-agnostic standard Gaussian as the importance sampling proposal, RA enables a finer-grained treatment for query-specific information and greatly improves the approximation fidelity; however, it is as expensive as softmax attention computationally with quadratic complexity, because the key-value statistics are different for each query, unlike the ones in RFAs.

[*] The majority of this work was done while the first author was interning at Bytedance. [1]Department of Computer Science, The University of Hong Kong [2]ByteDance Inc. [3]Shanghai Artificial Intelligence Laboratory. Correspondence to: Lin Zheng <linzheng@connect.hku.hk>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

[1]There are several variants of random feature maps that yield an unbiased estimate of the exponential kernel (Peng et al., 2021b; Choromanski et al., 2021). Nevertheless, RFAs still run a biased approximation to the whole softmax attention, since the softmax attention involves a ratio of these exponential kernels. Although the estimator is still consistent, the bias in question is elusive and might impair the approximation fidelity of random features.

Based on the analysis, one question naturally arises: *"Can we combine the expressiveness in RA and the efficiency in RFA to get the best of both worlds?"* To achieve that, we generalize the importance sampling formulation of RFA by adopting *multiple proposals*, each of which depends on different subsets of queries. We further apply multiple importance sampling (Veach & Guibas, 1995) and put together these proposals to approximate softmax attention adaptively for different queries, retaining the query-specific property of RA. Meanwhile, since these proposals are shared among all queries, we inherit the efficient computation reuse in RFA and achieve linear complexity. We refer to this efficient attention mechanism as LineAr Randomized Attention (LARA). Extensive experiments and analyses demonstrate that RA, as well as its linear variant LARA, significantly reduce the approximation error of RFAs. They improve RFAs by a substantial margin across various tasks, including image classification, video action recognition, machine translation, and so on, while retaining computational efficiency.

2. Background

2.1. Softmax Attention

Let $\mathbf{Q} \in \mathbb{R}^{N\times D}$, $\mathbf{K} \in \mathbb{R}^{M\times D}$ and $\mathbf{V} \in \mathbb{R}^{M\times D}$ denote the sets of N query vectors, M key and value vectors respectively. For each query $\mathbf{q}_n$, the softmax attention computes the following quantity,[2]

$$\mathsf{SoftmaxAttn}\left(\mathbf{q}_n, \mathbf{K}, \mathbf{V}\right) := \sum_{m=1}^{M} \frac{\exp(\mathbf{q}_n^\top \mathbf{k}_m)}{\sum_{m'=1}^{M}\exp(\mathbf{q}_n^\top \mathbf{k}_{m'})}\mathbf{v}_m^\top.$$

Intuitively, the softmax attention first computes the normalized similarity between the query and each key, which is then used to weight value vectors. In the case of self-attention in Transformers (Vaswani et al., 2017), we have $N = M$; as a result, such mechanism suffers from quadratic time and memory complexity due to the explicit computation of the similarity scores between all pairs of queries and keys.

2.2. Random Feature Attention

To reduce the computational complexity of softmax attention, recent work (Choromanski et al., 2021; Peng et al., 2021b) proposes to linearize exponential kernels via random feature methods (Rahimi & Recht, 2008). According to Bochner's theorem (Bochner, 2020), they re-write the exponential kernel $\exp\left(\mathbf{x}^\top\mathbf{y}\right)$ as the following expectation,

$$\exp(\mathbf{x}^\top\mathbf{y}) = \mathbb{E}_{\omega\sim\mathcal{N}(\omega;0,\mathbf{I})}\left[\xi(\mathbf{x},\omega)^\top\xi(\mathbf{y},\omega)\right], \qquad (1)$$

[2]We omit the commonly used scaling factor $1/\sqrt{d}$ for simplicity as it can be merged into the computation of queries or keys.

where $\xi(\cdot,\cdot): \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^l$, $l \geq 1$, is the *randomized mapping* transforming the input vector to a l-dimensional vector via a randomly drawn $\omega \sim \mathcal{N}(\omega; 0, \mathbf{I})$. A classical choice of randomized mapping is to let $\xi(\mathbf{x},\omega) = \exp\left(\frac{1}{2}\|\mathbf{x}\|^2\right)\left[\sin\left(\omega^\top\mathbf{x}\right), \cos\left(\omega^\top\mathbf{x}\right)\right]^\top$ (Rahimi & Recht, 2008; Peng et al., 2021b). In Performer (Choromanski et al., 2021), a scalar-valued positive randomized mapping $\xi(\mathbf{x},\omega) = \exp\left(\omega^\top\mathbf{x} - \frac{1}{2}\|\mathbf{x}\|^2\right)$ is used to improve the training stability. We base our model on the latter choice; other variants are discussed further in Appendix C. We use the term RFA and Performer interchangeably to refer to attention models with positive randomized mappings.

To estimate the expectation in Equation 1, we can draw multiple Monte Carlo samples[3] and compute the average such that $\exp(\mathbf{x}^\top\mathbf{y}) \approx \frac{1}{S}\sum_{s=1}^S \xi(\mathbf{x},\omega_s)^\top\xi(\mathbf{y},\omega_s)$. By substituting such approximation into the softmax attention, we obtain random feature attention (RFA; Choromanski et al., 2021; Peng et al., 2021b):

$$\begin{aligned}
&\frac{\sum_{m=1}^M \exp\left(\mathbf{q}_n^\top\mathbf{k}_m\right)\mathbf{v}_m^\top}{\sum_{m'=1}^M \exp\left(\mathbf{q}_n^\top\mathbf{k}_{m'}\right)} \\
&\approx \frac{\sum_{m=1}^M\sum_{s=1}^S \xi(\mathbf{q}_n,\omega_s)^\top\xi(\mathbf{k}_m,\omega_s)\mathbf{v}_m^\top}{\sum_{m'=1}^M\sum_{s=1}^S \xi(\mathbf{q}_n,\omega_s)^\top\xi(\mathbf{k}_{m'},\omega_s)} \\
&= \frac{\sum_{s=1}^S \xi(\mathbf{q}_n,\omega_s)^\top\sum_{m=1}^M \xi(\mathbf{k}_m,\omega_s)\mathbf{v}_m^\top}{\sum_{s=1}^S \xi(\mathbf{q}_n,\omega_s)^\top\sum_{m'=1}^M \xi(\mathbf{k}_{m'},\omega_s)} \qquad (2) \\
&:= \mathsf{RFA}\left(\mathbf{q}_n, \mathbf{K}, \mathbf{V}\right).
\end{aligned}$$

Thanks to the linearized formulation, one can first pre-compute the corresponding key-value statistics $\sum_{m=1}^M \xi(\mathbf{k}_m,\omega_s)\mathbf{v}_m^\top$ and $\sum_{m=1}^M \xi(\mathbf{k}_m,\omega_s)$ once, and then reuse them for each query. Consequently, it achieves linear complexity in both time and memory with respect to the sequence length.

2.3. Self-normalized Importance Sampling

Importance sampling (IS) is a general approach to approximating expectation $\mathbb{E}_{p(\omega)}\left[f(\omega)\right]$ when it is difficult to draw samples directly from $p(\omega)$. By sampling from a tractable *proposal distribution* $q(\omega)$ instead, IS forms the following estimate to correct the sampling bias,

$$\mathbb{E}_{p(\omega)}\left[f(\omega)\right] = \mathbb{E}_{q(\omega)}\left[\frac{p(\omega)}{q(\omega)}f(\omega)\right] \approx \frac{1}{S}\sum_{s=1}^S \frac{p(\omega_s)}{q(\omega_s)}f(\omega_s),$$

where $p(\omega)/q(\omega)$ is often referred to as *importance weights*. Given that $q(\omega)$ is positive whenever $p(\omega) \neq 0$, IS yields

[3]This sample average can also be written as $\boldsymbol{\phi}(\mathbf{x},\mathbf{w})^\top\boldsymbol{\phi}(\mathbf{y},\mathbf{w})$ with $\boldsymbol{\phi}(\mathbf{x},\mathbf{w}) := 1/\sqrt{S}[\xi(\mathbf{x},\omega_1),\ldots,\xi(\mathbf{x},\omega_S)]^\top \in \mathbb{R}^{lS}$. Here $\boldsymbol{\phi}(\cdot,\cdot)$ are conventionally referred to as *random features* (Rahimi & Recht, 2008). We spell out individual samples as it simplifies the analysis later.

an unbiased estimation. However, if the target density takes the form $p(\omega) = \tilde{p}(\omega)/Z$ and its normalizing constant is difficult to compute, IS would be intractable since it requires evaluating $p(\omega)$ explicitly. Self-normalized importance sampling (SNIS), a variant of IS estimators, mitigates this issue by taking the following form (Owen, 2013),

$$\mathbb{E}_{p(\omega)}\left[f(\omega)\right] = \frac{\mathbb{E}_{q(\omega)}\left[\frac{p(\omega)}{q(\omega)}f(\omega)\right]}{\mathbb{E}_{q(\omega)}\left[\frac{p(\omega)}{q(\omega)}\right]}$$
$$\approx \frac{\frac{1}{S}\sum_{s=1}^{S}\frac{1}{Z}\frac{\tilde{p}(\omega_s)}{q(\omega_s)}f(\omega_s)}{\frac{1}{S}\sum_{s=1}^{S}\frac{1}{Z}\frac{\tilde{p}(\omega_s)}{q(\omega_s)}} = \frac{\sum_{s=1}^{S}\frac{\tilde{p}(\omega_s)}{q(\omega_s)}f(\omega_s)}{\sum_{s=1}^{S}\frac{\tilde{p}(\omega_s)}{q(\omega_s)}}. \quad (3)$$

The name *self-normalized* comes from the fact that the importance weights $p(\omega)/q(\omega)$ are normalized. Albeit at the cost of introducing a bias, this method cancels out the normalizing constant Z at both nominator and denominator. SNIS often works well in practice.

3. Randomized Attention

In this section, we present an alternative view of RFA, revealing new insights of how RFA approximates the softmax attention. In particular, we show that RFA can be recast as a self-normalized importance sampler and its target expectation is exactly softmax attention (§3.1). This reformulation allows us to construct an unbiased estimator for softmax attention. We refer this unbiased estimation as randomized attention (§3.2).

3.1. RFA as Self-normalized Importance Sampling

Note that the formulation of RFA (Equation 2) and SNIS (§2.3) both take the form as a ratio of sample averages drawing from a tractable distribution. This resemblance motivates us to treat RFA as an SNIS estimator and reverse-engineer the target expectation $\mathbb{E}_{p(\omega)}\left[f(\omega)\right]$ that RFA approximates. For this to hold, the nominator and denominator in Equation 2 should define a regular importance sampling estimator and a valid importance weight up to some constant Z respectively. Formally, denoting $q(\omega) := \mathcal{N}(\omega; 0, \mathbf{I})$, for any $\omega_s \sim q(\omega)$ we have

$$\begin{cases} \frac{p(\omega_s)}{q(\omega_s)}f(\omega_s) = \frac{1}{Z}\xi(\mathbf{q}_n,\omega_s)^\top\sum_{m=1}^{M}\xi(\mathbf{k}_m,\omega_s)\mathbf{v}_m^\top, \\ \frac{p(\omega_s)}{q(\omega_s)} \quad\quad\;\; = \frac{1}{Z}\xi(\mathbf{q}_n,\omega_s)^\top\sum_{m=1}^{M}\xi(\mathbf{k}_m,\omega_s). \end{cases} \quad (4)$$

Solving this relation gives concise formulations for both $f(\omega)$ and $p(\omega)$ (see Appendix A for the proof):

Proposition 3.1. *Let $q(\omega) = \mathcal{N}(\omega; 0, \mathbf{I})$ be the proposal, $\xi(\mathbf{x},\omega) = \exp\left(\omega^\top\mathbf{x} - \frac{1}{2}\|\mathbf{x}\|^2\right)$ be the positive randomized mapping in Choromanski et al. (2021) and $\mathbb{E}_{p(\omega)}\left[f(\omega)\right]$ be the unknown target expectation. Given the relation specified in Equation 4, the distribution $p(\omega)$ is a Gaussian mixture with parametric component weights and means,*

$$p(\omega) = \sum_{m=1}^{M}\pi_m\mathcal{N}(\omega; \mathbf{q}_n + \mathbf{k}_m, \mathbf{I}), \quad (5)$$

where $\pi_m = \frac{\exp\left(\mathbf{q}_n^\top\mathbf{k}_m\right)}{\sum_{m'=1}^{M}\exp(\mathbf{q}_n^\top\mathbf{k}_{m'})}$ is the component weight. Besides, $f(\omega)$ is an attention-like aggregation function over value vectors, which computes the linearized similarity between queries and keys via randomized mappings,

$$f(\omega) = \frac{\xi(\mathbf{q}_n,\omega)^\top\sum_{m=1}^{M}\xi(\mathbf{k}_m,\omega)\mathbf{v}_m^\top}{\xi(\mathbf{q}_n,\omega)^\top\sum_{m'=1}^{M}\xi(\mathbf{k}_{m'},\omega)}. \quad (6)$$

From this perspective, for each query $\mathbf{q}_n$, RFA uses $\mathcal{N}(\omega; 0, \mathbf{I})$ as the proposal to perform self-normalized importance sampling for the following expectation,[4]

$$\mathbb{E}_{p_n(\omega)}[f_n(\omega)] = \mathbb{E}_{p_n(\omega)}\left[\frac{\xi(\mathbf{q}_n,\omega)^\top\sum_{m=1}^{M}\xi(\mathbf{k}_m,\omega)\mathbf{v}_m^\top}{\xi(\mathbf{q}_n,\omega)^\top\sum_{m'=1}^{M}\xi(\mathbf{k}_{m'},\omega)}\right].$$

This re-formulation offers alternative viewpoints to understand the approximation quality of RFA. It is straightforward to see that RFA is a biased (but consistent) estimator due to the self-normalization (Owen, 2013). In addition, RFA may exhibit large bias and variance since it only uses a standard Gaussian proposal, which is far away from the underlying input-dependent mixture $p_n(\omega)$. These may explain its inferior performance and slow convergence observed in previous studies (Patrick et al., 2021; Tay et al., 2021b).

3.2. Randomized Attention

The analysis above further implies that the softmax attention itself can be formulated as an expectation.

Proposition 3.2. *Let $p_n(\omega)$ and $f_n(\omega)$ be defined by Equation 5 and Equation 6 respectively. Then for softmax attention we have*

$$\mathsf{SoftmaxAttn}(\mathbf{q}_n, \mathbf{K}, \mathbf{V}) = \mathbb{E}_{p_n(\omega)}\left[f_n(\omega)\right]. \quad (7)$$

The detailed proof is in Appendix B. As a result, RFA can be viewed as using importance sampling to estimate softmax attention. Alternatively, one can directly sample from $p_n(\omega)$ to construct an *unbiased* estimate of the softmax attention,

$$\begin{aligned} &\mathsf{SoftmaxAttn}(\mathbf{q}_n, \mathbf{K}, \mathbf{V}) \\ &\approx \frac{1}{S}\sum_{s=1}^{S}\frac{\xi(\mathbf{q}_n,\omega_s)^\top\sum_{m=1}^{M}\xi(\mathbf{k}_m,\omega_s)\mathbf{v}_m^\top}{\xi(\mathbf{q}_n,\omega_s)^\top\sum_{m'=1}^{M}\xi(\mathbf{k}_{m'},\omega_s)} \\ &:= \mathsf{RA}\left(\mathbf{q}_n, \mathbf{K}, \mathbf{V}\right) \end{aligned}$$

[4]Here we add the subscript to emphasize that both $f_n(\omega)$ and $p_n(\omega)$ is specific to a particular query $\mathbf{q}_n$.

with $\omega_1, \ldots, \omega_S \sim p_n(\omega)$. To the best of our knowledge, this is the first kernel linearization estimator that approximates the whole softmax attention, instead of just exponential kernels, in an *unbiased* manner. We refer to this estimator as *randomized attention* (RA), since it computes attention-like aggregations but via randomized mappings.

Intuitively, RA constructs the randomized mapping by sampling from the contextual distribution $p_n(\omega)$, which promotes ω in the vicinity of the resultant of current queries and keys. Aware of locations of query-key pairs, ω is likely to describe their similarity better than input-agnostic ones as in RFA. In addition, each query position n in RA induces an exclusive distribution p_n, which makes the randomized mapping adaptive to each query. This allows the model to process query information at a finer-grained level and thus achieves higher approximation fidelity (see §5 for empirical validation). Nevertheless, the use of query-specific modeling requires to draw a different set of samples for different queries. As a result, the mapped key statistics $\xi(\mathbf{k}_m, \omega)$ will be different for different queries, which prevents reusing the computation and results in $\mathcal{O}(MN)$ complexity, rendering it less applicable in approximating softmax attention in practice.

This is in sharp contrast to RFA. RFA uses the same proposal $\mathcal{N}(\omega; 0, \mathbf{I})$ for all queries, and thus the modeling power is greatly reduced since the standard Gaussian would capture neither contextual information nor the inherent variations among queries. The advantage of the shared proposal is that it enables efficient computation reuse of key-value statistics (Equation 2), as the same randomized mapping is reused across queries. This property accounts for RFA's linear complexity.

4. Linear Complexity Randomized Attention

In this section, we propose an improved estimator of softmax attention to combine both the expressiveness of RA and the efficiency of RFA. Motivated by the difference between RA and RFA, we generalize the importance sampling formulation of RFA by adopting *multiple* proposals. This strategy not only captures query information at a finer-grained level, but also allows the model to estimate softmax attention in a query-specific manner (§4.1). We further show that computation reuse in RFA can be achieved, which leads to linear complexity computation with the help of self-normalized importance sampling (§4.2).

4.1. Importance Sampling with Multiple Proposals

As discussed in §3.2, both RA and RFA aim to estimate the expectation $\mathbb{E}_{p_n(\omega)}\left[f_n(\omega)\right]$ (Equation 7). The main difference between RA and RFA is that RA samples from a distinct distribution for each query, while RFA uses the same proposal distribution for all queries. To get the best of both worlds, we propose to adopt a set of C ($C \ll N$) proposal distributions $\{q_c(\omega)\}_{c=1}^{C}$ for our estimation, each of which depends on a *subset* of queries (see Appendix G.3.1 for the detailed discussion on parameterizing these proposals).

This strategy not only enables a finer-grained treatment for query information, but also allows the model to estimate softmax attention in a query-specific way, which is the key advantage of RA. To be specific, since there are several proposals available for each query, and these proposals may provide complementary information to each other, we could combine them by invoking multiple importance sampling (MIS; Veach & Guibas, 1995). For each query, the MIS estimate takes the following form,[5]

$$\mathbb{E}_{p_n(\omega)}\left[f_n(\omega)\right] \approx \sum_{c=1}^{C} \alpha_{nc}(\omega_c) \frac{p_n(\omega_c)}{q_c(\omega_c)} f_n(\omega_c) \qquad (8)$$

where $\omega_c \sim q_c(\omega)$ for $c = 1, \ldots, C$ and $\{\alpha_{nc}(\cdot)\}_{c=1}^{C}$ are *weighting functions*. The MIS estimator is unbiased (Veach & Guibas, 1995) if $\sum_{c=1}^{C} \alpha_{nc}(\omega) = 1$ for any ω (see the proof in Appendix F).[6] Intuitively, MIS first computes individual importance sampling estimates with each proposal, which are averaged together according to the *query-specific* weighting functions.

Ideally, the n-th set of weighting functions $\{\alpha_{nc}(\cdot)\}_{c=1}^{C}$ should specialize in processing the n-th query. To accomplish this goal, we expect weighting functions to be optimal (i.e., minimize the estimation variance) for the corresponding query. Optimal weighting functions takes the following form (detailed derivation can be found in Appendix D),

$$\begin{aligned}\alpha^*_{nc}(\omega_c) = &\frac{q_c(\omega_c)}{\sum_{c'=1}^{C} q_{c'}(\omega_c)} + \\ &q_c(\omega_c)\left(r_{nc}(\omega_c) - \sum_{c=1}^{C} \frac{q_c(\omega_c)}{\sum_{c'=1}^{C} q_{c'}(\omega_c)} r_{nc}(\omega_c)\right).\end{aligned}$$

Here $r_{nc}(\cdot)$ is roughly proportional to the closeness between $q_c(\cdot)$ and the query-specific optimal proposal. Intuitively, the optimal weighting function consists of two terms. The first term is query-agnostic and the second term is a query-specific correction. The correction term is defined by the difference between $r_{nc}(\cdot)$ and its average weighted by $q_c(\cdot)$; consequently, if $r_{nc}(\cdot)$ is large, the correction term will be positive, driving the weight of the c-th proposal to be higher and vice versa.

[5]Here we assume only one sample is drawn from each proposal distribution. A more general treatment would allow arbitrary numbers of samples to be drawn from each proposal.

[6]Strictly speaking, for the MIS estimator to be unbiased, we additionally need the weighting functions to be zero for any ω such that $p_n(\omega) = 0$, although this holds trivially in our setting.

In most cases, it is intractable to apply optimal weighting functions, since the closed form of $r_{nc}(\cdot)$ is not available. We therefore approximate the optimal weighting functions by the following form,

$$\alpha_{nc}(\omega_c) = \frac{q_c(\omega_c)}{\sum_{c'=1}^{C} q_{c'}(\omega_c)} + r'_{nc} - \frac{1}{C}\sum_{c=1}^{C} r'_{nc}, \tag{9}$$

where r'_{nc} measures the degree of the proposal q_c favoring the n-th query. For tractability, we implement r'_{nc} as the normalized similarity between the n-th query and the representation of the c-th query subset. We also decouple the computation between proposal densities $q_c(\omega)$ and r'_{nc}, so that contributions from query-agnostic and query-specific terms can be independent of each other (see Appendix G.3.2 for more details and ablations). Note that Equation 9 still ensures unbiasedness (or consistency) of MIS estimation due to $\sum_{c=1}^{C} \alpha_{nc}(\omega) = 1$.

4.2. Achieving Linear Time and Space Complexity

According to our MIS estimator (Equation 8), the key-value statistics under each proposal can be pre-computed once and then reused for all queries. This implies the computation reuse in RFA is achievable and so as the linear complexity.

The only problem left now is that the MIS estimator still requires explicitly evaluating the density $p_n(\omega)$ for each query (Equation 5), which exhibits quadratic complexity. This is because $p_n(\omega)$ is a Gaussian mixture with M components, incurring $\mathcal{O}(NM)$ computations in total. We show that a self-normalized version of MIS allows us to further reduce the complexity to be linear. According to Proposition 3.1 (and Equation 15 in Appendix A), the mixture density $p_n(\omega)$ can be equivalently expressed as

$$p_n(\omega) = \frac{\mathcal{N}(\omega; 0, \mathbf{I})\xi(\mathbf{q}_n, \omega)^\top \sum_{m=1}^{M} \xi(\mathbf{k}_m, \omega)}{\sum_{m'=1}^{M} \exp\left(\mathbf{q}_n^\top \mathbf{k}_{m'}\right)} := \frac{\tilde{p}_n(\omega)}{Z_p}.$$

Our key observation is that now the numerator contains a linearized dot product of randomized mappings, which can be pre-computed and reused for all queries, while the denominator is similar to the normalizing constant in regular softmax attention and can only be computed in quadratic time. Fortunately, the denominator can be canceled out if we adopt the *self-normalized* estimator (see §2.3),

$$\begin{aligned}\mathbb{E}_{p_n(\omega)}\left[f_n(\omega)\right] &\approx \frac{\sum_{c=1}^{C} \alpha_{nc}(\omega_c)\frac{\tilde{p}_n(\omega_c)}{q_c(\omega_c)} f_n(\omega_c)}{\sum_{c=1}^{C} \alpha_{nc}(\omega_c)\frac{\tilde{p}_n(\omega_c)}{q_c(\omega_c)}} \\ &:= \mathsf{LARA}\left(\mathbf{q}_n, \mathbf{K}, \mathbf{V}\right).\end{aligned} \tag{10}$$

The resulting estimator is consistent and runs with linear complexity, similar to RFA. We name it linear randomized attention (LARA). See Algorithm 3 in Appendix G.3 for an algorithmic sketch of LARA.

4.3. Discussion: RFA, RA, and LARA

LARA defines a flexible framework to bridge RFA and RA. To delineate the connection between RFA and LARA, we find LARA can be further rewritten as (see Appendix E for the derivation)

$$\begin{aligned}&\mathsf{LARA}\left(\mathbf{q}_n, \mathbf{K}, \mathbf{V}\right) \\ &\quad= \frac{\sum_{c=1}^{C} \alpha'_{nc}(\omega_c)\xi(\mathbf{q}_n, \omega_c)^\top \sum_{m=1}^{M} \xi(\mathbf{k}_m, \omega_c)\mathbf{v}_m^\top}{\sum_{c=1}^{C} \alpha'_{nc}(\omega_c)\xi(\mathbf{q}_n, \omega_c)^\top \sum_{m=1}^{M} \xi(\mathbf{k}_m, \omega_c)},\end{aligned}$$

where $\omega_c \sim q_c(\omega)$ for $c = 1, \ldots, C$ and $\alpha'_{nc}(\omega_c) := \alpha_{nc}(\omega_c)\mathcal{N}(\omega_c; 0, \mathbf{I})/q_c(\omega_c)$. Comparing to the formulation of RFA (Equation 2), we see that RFA is a special case of LARA if we set all proposals to $\mathcal{N}(\omega; 0, \mathbf{I})$ and all $\alpha_{nc}(\cdot)$ to constant functions. On the other hand, LARA is equivalent to RA if we remove the use of self-normalization, set $\alpha_{nc}(\omega) = \delta_{nc}$ [7] and maintain N proposals, each of which takes the same form of $p_n(\omega)$ (Equation 5). With general proposals and weighting functions, LARA approximates softmax attention in a query-specific manner as in RA while achieving linear complexity as in RFA, effectively combining the advantages of both estimators.

5. Experiments

In this section, we conduct extensive experiments across various domains to verify the effectiveness of linear randomized attention. Firstly, we start with an experiment to assess the approximation error of different random feature based methods (§5.1). We then perform a number of experiments on various data modalities, including image classification (§5.2), video action recognition (§5.3), machine translation (§5.4), and long sequence modeling on Long Range Arena benchmark (Appendix I.2). Additional details as well as ablation studies can be found in Appendices H and I. The implementation details of RA, Performer (RFA) and LARA are provided in Appendix G.

5.1. Experiments on the Approximation Quality

We conduct a preliminary experiment to assess the approximation fidelity of different random feature methods (details are deferred to Appendix H.1). In particular, we consider vision transformers (ViT; Dosovitskiy et al., 2021; Touvron et al., 2021), keep $\mathbf{Q}, \mathbf{K}$ and $\mathbf{V}$ the same across attention variants, and compute the Mean Squared Error (MSE) between the outputs of true softmax attention and its approximations. We use the `ImageNet1k` validation set (see more details in §5.2) as the input data and report MSE results averaged over all images. Figure 1 shows the results with respect to the number of random samples under different sequence

[7]That is, weighting functions now become the Kronecker delta function, where $\alpha_{nc}(\omega) = 1$ if $n = c$ and 0 otherwise.

Table 1. Classification results on `ImageNet1k` dataset with DeiT architectures under different attention mechanisms. "*-8" denotes the corresponding attention method with patch size 8, resulting in longer sequence with length 784; N denotes the sequence length.

Model	Complexity	DeiT-Tiny		DeiT-Small	
		# Param.	Top-1 Acc.	# Param.	Top-1 Acc.
Performer	$\mathcal{O}(N)$	5.7M	65.92	22.0M	74.29
Performer-8	$\mathcal{O}(N)$	5.7M	67.79	22.0M	74.57
LARA	$\mathcal{O}(N)$	5.8M	71.48	22.2M	79.48
LARA-8	$\mathcal{O}(N)$	5.8M	**74.16**	22.2M	**80.62**
RA	$\mathcal{O}(N^2)$	5.7M	71.86	22.0M	80.04
Softmax	$\mathcal{O}(N^2)$	5.7M	72.20	22.0M	79.90

lengths. We observe that RFA (Performer) soon plateaus at large approximation error and does not improve even with more samples, possibly due to low sample efficiency. On the other hand, LARA exhibits much lower MSE than Performer and the approximation error continually decreases as the number of samples increases. As for RA, it achieves the lowest MSE among these three methods. This clearly indicates that increasing the model's resolution over query positions as in LARA and RA is more effective in improving approximation quality, compared to simply increasing the sample size from the same distribution (as in Performer).

5.2. Image Classification

For image classification, we conduct our experiment on the `ImageNet1k` benchmark (Deng et al., 2009), which consists of approximately 1,280K/50K images over 1,000 classes for training/validation splits respectively. We apply our attention mechanism to different vision transformer (ViT) architectures (Dosovitskiy et al., 2021), including DeiT (Touvron et al., 2021) and pyramid vision transformers v2 (PVTv2; Wang et al., 2021a;b). The former architecture adopts standard transformer layers with regular softmax attention and receives sequence with length 196 by default; while the latter processes much longer image sequences, which is therefore more suitable to evaluate the scalability of various efficient attention. More model and training details can be found in Appendix H.2.

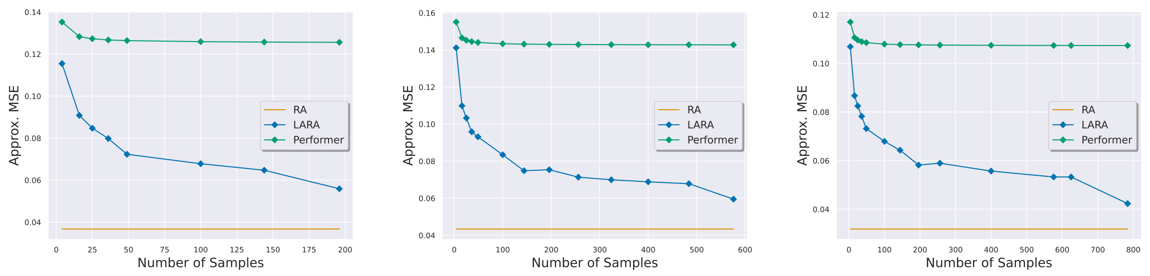
Results on DeiT. The comparison among different random feature based methods on DeiT model is demonstrated in Table 1. Consistent with previous studies (Zheng et al., 2021), Performer (RFA) incurs a significant performance drop due to its limited modeling capacity. Its unbiased counterpart, RA, performs much better than Performer and even slightly outperforms exact softmax attention under larger model sizes. This empirically validates the expressiveness of unbiasedness in approximating softmax attention. LARA achieves a good trade-off between Performer and RA. It enjoys linear complexity as Performer but performs substantially better. On the other hand, we note that a linear

Table 2. Classification results on `ImageNet1k` dataset compared with state-of-the-art model architectures.

Model	# Param.	FLOPs	Top-1 Acc.
PVT-v1-T (Wang et al., 2021a)	13.2M	2.1G	75.1
SOFT-T (Lu et al., 2021)	13.1M	1.9G	79.3
RegionViT-T (Chen et al., 2021b)	13.8M	2.4G	**80.4**
PVT-v2-b1 (SRA)	14.0M	2.1G	78.7
PVT-v2-b1 + Performer	12.1M	2.5G	77.3
PVT-v2-b1 + LARA	13.7M	2.3G	79.6
PVT-v1-S (Wang et al., 2021a)	24.5M	3.8G	79.8
DeiT-S (Touvron et al., 2021)	22.1M	4.6G	79.9
RegNetY-4G (Radosavovic et al., 2020)	21.0M	4.0G	80.0
Swin-T (Liu et al., 2021)	28.3M	4.5G	81.3
CvT-13 (Wu et al., 2021)	20.0M	4.5G	81.6
Twins-SVT-S (Chu et al., 2021)	24.0M	2.8G	81.7
SOFT-S (Lu et al., 2021)	24.1M	3.3G	82.2
Focal-T (Yang et al., 2021)	29.1M	4.9G	82.2
ViL-S (Zhang et al., 2021)	24.6M	4.9G	82.4
PVT-v2-b2 (SRA)	25.4M	4.0G	82.1
PVT-v2-b2 + Performer	21.1M	4.9G	81.0
PVT-v2-b2 + LARA	22.4M	4.5G	**82.6**
PVTv1-M (Wang et al., 2021a)	44.2M	6.7G	81.2
RegNetY-8G (Radosavovic et al., 2020)	39.0M	8.0G	81.7
CvT-21 (Wu et al., 2021)	32.0M	7.1G	82.5
SOFT-M (Lu et al., 2021)	45.0M	7.2G	82.9
RegionViT-M (Chen et al., 2021b)	42.0M	7.9G	83.4
ViL-M (Zhang et al., 2021)	39.7M	9.1G	83.5
PVT-v2-b3 (SRA)	45.2M	6.9G	83.3
PVT-v2-b3 + Performer	36.0M	8.2G	82.4
PVT-v2-b3 + LARA	39.9M	7.7G	**83.6**
PVTv1-L (Wang et al., 2021a)	61.4M	9.8G	81.7
RegNetY-16G (Radosavovic et al., 2020)	84.0M	16.0G	82.9
Swin-S (Liu et al., 2021)	50.0M	8.7G	83.0
SOFT-L (Lu et al., 2021)	64.1M	11.0G	83.1
Focal-S (Yang et al., 2021)	51.1M	9.1G	83.5
ViL-B (Zhang et al., 2021)	55.7M	13.4G	83.7
RegionViT-B (Chen et al., 2021b)	73.8M	13.6G	83.8
PVT-v2-b4 (SRA)	62.6M	10.1G	83.6
PVT-v2-b4 + Performer	48.6M	11.9G	82.7
PVT-v2-b4 + LARA	54.5M	11.3G	**84.0**

complexity variant enables the transformer model to scale to much longer sequences, which is often prohibitive for traditional softmax attention but delivers better predictive performance (El-Nouby et al., 2021). We thus train Performer and LARA with 8×8 image patches (resulting in sequence length 784) with all other settings unchanged. As shown in Table 1, increasing the sequence length (suffixed with "-8") consistently boosts model performance. However, LARA benefits from longer sequences much more significantly than Performer and outperforms softmax attention by a large margin. This indicates the potential modeling power of our framework for long sequences. Also see Appendix I.1 for additional experiments and ablations.

Results on PVTv2. We then apply our method to the strong baseline PVTv2 and compare it against recent state-of-the-art model architectures. As presented by Table 2, we observe although replacing spatial reduction attention (SRA; details in §H.2) with Performer leads to inferior performance, LARA brings a consistent performance gain over



(a) Sequence length $N = 196$. (b) Sequence length $N = 576$. (c) Sequence length $N = 784$.

Figure 1. Mean Squared Error (MSE) between the true softmax attention and different approximation methods under different numbers of samples (lower is better). Results are evaluated on ViTs under typical settings of sequence length, including 196, 576 and 784. Note that we only draw 1 sample in RA estimation so that the curve of RA is constant.

vanilla SRA with much fewer model parameters. In addition, PVTv2 with LARA even performs highly competitive with state-of-the-art architectures across various model sizes, without introducing other inductive biases (such as locality). This implies the superior modeling capacity of LARA compared to SRA and Performer.

5.3. Video Action Recognition

In this section, we test our method on video action recognition with video transformers. We consider two standard datasets: (1) Kinetics-400 (K400; Kay et al., 2017), which contains 238,574 videos for training and 19,877 for evaluation at the time of writing and (2) Something-something-v2 (SSv2; Goyal et al., 2017), consisting of around 168K/25K videos of 174 classes for training/validation splits respectively. We base our model on the Motionformer architecture (Patrick et al., 2021) and follow their training and evaluation protocol; more details can be found in Appendix H.3.

Table 3 reports the top-1 classification accuracy for both `K400` and `SSv2` datasets. We see that RA still achieves the best performance among attention approximations albeit falling behind the exact softmax attention. Since Motionformer is pretrained on images with *softmax attention*, this gap is likely introduced by employing a different attention mechanism during training the model further on video datasets. Besides, LARA outperforms Performer and Nyströmformer (Xiong et al., 2021) by a large margin on both `K400` and `SSv2` datasets. Although achieving strong performance, Orthoformer (Patrick et al., 2021) runs much slower (roughly 3× or more) than other attention variants due to its sequential nature. As a result, LARA achieves better trade-offs than these baselines between predictive accuracy and efficiency.

Table 3. Video action recognition accuracy on `K400` and `SSv2` datasets with different attention mechanisms. N denotes the spatial sequence length.

Model	Complexity	Acc. (%) on `K400`	Acc. (%) on `SSv2`
Nyströmformer	$\mathcal{O}(N)$	76.5	61.7
Orthoformer	$\mathcal{O}(N)$	77.8	64.7
Performer	$\mathcal{O}(N)$	72.1	53.1
LARA	$\mathcal{O}(N)$	77.5	63.7
RA	$\mathcal{O}(N^2)$	78.2	64.9
Exact Motionformer	$\mathcal{O}(N^2)$	79.2	66.5

5.4. Machine Translation

In this section, we conduct experiments on WMT14 EN–DE machine translation benchmark (Bojar et al., 2014) to evaluate the performance of our model under various sequence lengths. We follow Vaswani et al. (2017) and Ott et al. (2018) to preprocess this dataset, resulting in about 4.5M/3K/3K sentences pairs for training/validation/testing splits respectively. We adopt the standard transformer base architecture (Vaswani et al., 2017) and replace encoder self-attention with efficient attention variants. More detailed configurations are deferred to Appendix H.4.

Table 4 presents the test BLEU scores under different attention mechanisms. Since this dataset consists mostly of short sentences, we set the number of samples to be relatively smaller. However, the training of Performer is quite unstable and a larger number of samples is required to mitigate this issue. Besides, we observe a similar trend that replacing the standard softmax attention with Performer leads to a significant performance drop, while increasing the number of samples does not improve the translation quality. RA, on the other hand, even outperforms softmax attention by over 0.3 BLEU score, clearly demonstrating the modeling capacity of unbiased approximations. LARA reaches performance close to softmax attention while runs with the same complexity as Performer; compared to other attention variants, LARA outperforms both Linformer (Wang et al., 2020) and ABC (Peng et al., 2021a) while obtaining similar BLEU scores to Nyströmformer (Xiong et al., 2021). This indicates RA and LARA are also capable of modeling natural

Table 4. Test BLEU scores on WMT14 EN-DE dataset under different attention mechanisms. For brevity, "# samples" denotes either the number of samples or landmarks involved in different attention variants; – indicates the model does not converge during training; n.a. denotes not applicable.

Model	# samples	# Param.	BLEU
Softmax	n.a.	60.92M	27.5
ABC	16	60.93M	25.4
	32	60.94M	25.6
	64	60.95M	26.0
Linformer	16	60.92M	17.4
	32	61.31M	23.0
	64	61.70M	23.7
Nyströmformer	16	60.92M	25.1
	32	60.92M	26.8
	64	60.92M	26.8
Performer	64	60.92M	–
	128	60.92M	23.5
	256	60.92M	23.7
	512	60.92M	23.3
LARA	16	60.96M	26.4
	32	60.96M	26.8
	64	60.96M	27.0
RA	n.a.	60.92M	**27.8**

language, which is typically hierarchically structured.

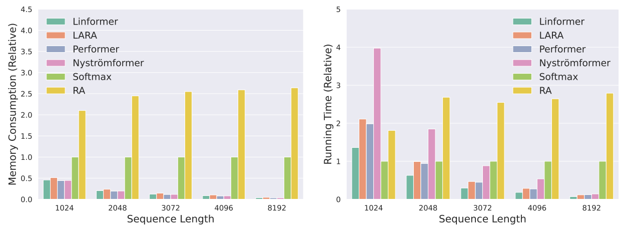
5.5. Analysis on Time and Memory Consumption

To evaluate the empirical efficiency of various attention methods, we conduct a simulation on a standard transformer architecture and report the running time and memory consumption under different sequence lengths. The detailed setup can be found in Appendix H.5. As shown in Figure 2 (and Table 6 in Appendix H.5 for exact statistics), we note that RA runs twice (or more) as slow as ordinary softmax attention with about $2.5\times$ memory consumption. This is as expected since RA needs to first compute full softmax probabilities to sample from p_n, and then compute f_n, both of which take a similar amount of computation to softmax attention. Nevertheless, its efficient variant LARA runs as fast as Performer with marginally increased memory usage. As for another baseline Nyströmformer (Xiong et al., 2021), which we found is a strong baseline and is used across experiments, it runs much slower than other variants at relatively short sequence lengths (e.g., less than 8192). Overall, the comparison result validates that LARA achieves a good balance between efficiency and expressiveness.

6. Related Work

Transformer models (Vaswani et al., 2017) are difficult to scale to long sequences due to the quadratic time and space complexity of self-attention mechanisms. Recently, a significantly large number of approaches have been proposed to improve the efficiency of attention mechanisms. A widely adopted paradigm is to utilize sparse attention, where each query is limited to only attend a subset of tokens. Such sparse attentive patterns can be pre-defined, such as sliding windows (Beltagy et al., 2020) or block-wise local chunks (Liu* et al., 2018; Parmar et al., 2018; Child et al., 2019; Ainslie et al., 2020; Zaheer et al., 2020; Liu et al., 2021); alternatively, the model can adaptively select tokens to take into account. This can be done via a trainable top-k selecting operator (Pietruszka et al., 2020), learnable hash functions (Kitaev et al., 2020; Daras et al., 2020), clustering with K-Means (Vyas et al., 2020; Roy et al., 2021) or grouping tokens with a differentiable sorting module (Tay et al., 2020a). More recently, Combiner (Ren et al., 2021) is proposed to apply the sparse mechanism to factorize the softmax probability distribution so that the resulting approximation runs with sub-quadratic time but achieves full attention capacity.

Low-rank approximations to the softmax attention also received considerable interest. For instance, the Nyström method can be adopted to approximate the softmax attention map by a sub-sampled matrix (Xiong et al., 2021). Another approach is the kernel linearization, which aims to decompose the exponential kernel into a dot product of feature maps. Such feature maps can be randomized that yield unbiased estimates of exponential kernels (Choromanski et al., 2021; Peng et al., 2021b), or deterministic that enjoy better training convergence (Katharopoulos et al., 2020; Kasai et al., 2021b; Schlag et al., 2021). Alternatively, one can use a learnable matrix (including Linformer (Wang et al., 2020) and ABC (Peng et al., 2021a)) or other downsampling operations (Dai et al., 2020; Wang et al., 2021a;b) to project the key-value pairs into fixed-length sequences. Besides, a set of auxiliary points can also be incorporated to cache the information from the long sequence via an attention mechanism, which is adopted in LUNA (Ma et al., 2021), Set transformer (Lee et al., 2019) and Perceiver (Jaegle et al., 2021a;b). Our work falls into the category of kernel lin-



(a) Memory consumption. (b) Running time.

Figure 2. Empirical memory consumption (left) and running time (right) of different attention mechanisms under different sequence lengths. Metrics are measured relative to the softmax attention.

earization methods, but in contrast to previous works, we propose an unbiased estimation for the *whole* softmax attention, which has not been explored and is orthogonal to previous works.

Recent studies also consider combining both the sparse and low-rank bias to achieve better approximation (Nguyen et al., 2021; Zhu et al., 2021; Chen et al., 2021a), or replace the softmax attention with other token-mixing mechanisms (Lee-Thorp et al., 2021; Lu et al., 2021; Chen et al., 2021d; Tay et al., 2021a). We refer readers to Tay et al. (2020b; 2021b); Lin et al. (2021) for a more detailed review on advances in the topic of efficient attention.

7. Conclusion

In this paper, we revisit the recently proposed random feature methods for approximating the softmax attention. By recasting RFA as self-normalized importance samplers, we identify an elusive bias in its approximation process. Built on this finding, we propose the unbiased estimation, called randomized attention (RA), which constructs positive random features via query-specific distributions. We then develop a novel linear complexity self-attention mechanism called linear randomized attention (LARA), which combines the expressiveness in RA and the efficiency in RFA. Extensive experiments demonstrate the effectiveness of RA and LARA, across various domains.

Acknowledgements

We thank Jianbo Yuan, Xiang Gao, Xiujun Li, Yanghua Peng, Ding Zhou, and Ruofan Ding for helpful discussions and feedback on early drafts of this paper. This research was supported in part by the joint research scheme of the National Natural Science Foundation of China (NSFC) and the Research Grants Council (RGC) under grant number N_HKU714/21.

References

Ainslie, J., Ontanon, S., Alberti, C., Cvicek, V., Fisher, Z., Pham, P., Ravula, A., Sanghai, S., Wang, Q., and Yang, L. Etc: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 268–284, 2020.

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

Berman, M., Jégou, H., Vedaldi, A., Kokkinos, I., and Douze, M. Multigrain: a unified image embedding for classes and instances. *arXiv preprint arXiv:1902.05509*, 2019.

Bochner, S. *Harmonic analysis and the theory of probability*. University of California press, 2020.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., et al. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pp. 12–58, 2014.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pp. 213–229. Springer, 2020.

Chen, B., Dao, T., Winsor, E., Song, Z., Rudra, A., and Ré, C. Scatterbrain: Unifying sparse and low-rank attention. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021a.

Chen, C.-F., Panda, R., and Fan, Q. Regionvit: Regional-to-local attention for vision transformers. *arXiv preprint arXiv:2106.02689*, 2021b.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021c.

Chen, Y., Zeng, Q., Ji, H., and Yang, Y. Skyformer: Remodel self-attention with gaussian kernel and nystr\" om method. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021d.

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

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Chu, X., Tian, Z., Wang, Y., Zhang, B., Ren, H., Wei, X., Xia, H., and Shen, C. Twins: Revisiting the design of spatial attention in vision transformers. *arXiv preprint arXiv:2104.13840*, 2021.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 702–703, 2020.

Dai, Z., Lai, G., Yang, Y., and Le, Q. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *Advances in neural information processing systems*, 33:4271–4282, 2020.

Daras, G., Kitaev, N., Odena, A., and Dimakis, A. G. Smyrf-efficient attention using asymmetric clustering. *Advances in Neural Information Processing Systems*, 33, 2020.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

El-Nouby, A., Touvron, H., Caron, M., Bojanowski, P., Douze, M., Joulin, A., Laptev, I., Neverova, N., Synnaeve, G., Verbeek, J., et al. Xcit: Cross-covariance image transformers. *arXiv preprint arXiv:2106.09681*, 2021.

Fan, H., Li, Y., Xiong, B., Lo, W.-Y., and Feichtenhofer, C. Pyslowfast. https://github.com/facebookresearch/slowfast, 2020.

Goyal, R., Ebrahimi Kahou, S., Michalski, V., Materzynska, J., Westphal, S., Kim, H., Haenel, V., Fruend, I., Yianilos, P., Mueller-Freitag, M., et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pp. 5842–5850, 2017.

Hoffer, E., Ben-Nun, T., Hubara, I., Giladi, N., Hoefler, T., and Soudry, D. Augment your batch: Improving generalization through instance repetition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8129–8138, 2020.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016.

Jaegle, A., Borgeaud, S., Alayrac, J.-B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021a.

Jaegle, A., Gimeno, F., Brock, A., Vinyals, O., Zisserman, A., and Carreira, J. Perceiver: General perception with iterative attention. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664. PMLR, 18–24 Jul 2021b. URL https://proceedings.mlr.press/v139/jaegle21a.html.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Kasai, J., Pappas, N., Peng, H., Cross, J., and Smith, N. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=KpfasTaLUpq.

Kasai, J., Peng, H., Zhang, Y., Yogatama, D., Ilharco, G., Pappas, N., Mao, Y., Chen, W., and Smith, N. A. Finetuning pretrained transformers into rnns. *arXiv preprint arXiv:2103.13076*, 2021b.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kitaev, N., Kaiser, L., and Levskaya, A. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

Kondapaneni, I., Vevoda, P., Grittmann, P., Skřivan, T., Slusallek, P., and Křivánek, J. Optimal multiple importance sampling. *ACM Trans. Graph.*, 38 (4), jul 2019. ISSN 0730-0301. doi: 10.1145/3306346.3323009. URL https://doi.org/10.1145/3306346.3323009.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

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

Lee-Thorp, J., Ainslie, J., Eckstein, I., and Ontanon, S. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*, 2021.

Lin, T., Wang, Y., Liu, X., and Qiu, X. A survey of transformers. *arXiv preprint arXiv:2106.04554*, 2021.

Linsley, D., Kim, J., Veerabadran, V., Windolf, C., and Serre, T. Learning long-range spatial dependencies with horizontal gated recurrent units. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 152–164, 2018.

Liu*, P. J., Saleh*, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hyg0vbWC-.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10012–10022, October 2021.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Lu, J., Yao, J., Zhang, J., Zhu, X., Xu, H., Gao, W., Xu, C., Xiang, T., and Zhang, L. Soft: Softmax-free transformer with linear complexity. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Ma, X., Kong, X., Wang, S., Zhou, C., May, J., Ma, H., and Zettlemoyer, L. Luna: Linear unified nested attention. *arXiv preprint arXiv:2106.01540*, 2021.

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011.

Nangia, N. and Bowman, S. R. Listops: A diagnostic dataset for latent tree learning. *arXiv preprint arXiv:1804.06028*, 2018.

Nguyen, T., Suliafu, V., Osher, S., Chen, L., and Wang, B. Fmmformer: Efficient and flexible transformer via decomposed near-field and far-field attention. *Advances in Neural Information Processing Systems*, 34, 2021.

Ott, M., Edunov, S., Grangier, D., and Auli, M. Scaling neural machine translation. *arXiv preprint arXiv:1806.00187*, 2018.

Owen, A. B. *Monte Carlo theory, methods and examples*. 2013.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. Image transformer. In *International Conference on Machine Learning*, pp. 4055–4064. PMLR, 2018.

Patrick, M., Campbell, D., Asano, Y., Misra, I., Metze, F., Feichtenhofer, C., Vedaldi, A., and Henriques, J. F. Keeping your eye on the ball: Trajectory attention in video transformers. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=mfQxdSMWOF.

Peng, H., Kasai, J., Pappas, N., Yogatama, D., Wu, Z., Kong, L., Schwartz, R., and Smith, N. A. Abc: Attention with bounded-memory control. *arXiv preprint arXiv:2110.02488*, 2021a.

Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N., and Kong, L. Random feature attention. In *International Conference on Learning Representations*, 2021b.

Pietruszka, M., Borchmann, Ł., and Garncarek, Ł. Sparsifying transformer models with trainable representation pooling. *arXiv preprint arXiv:2009.05169*, 2020.

Radev, D. R., Muthukrishnan, P., Qazvinian, V., and Abu-Jbara, A. The acl anthology network corpus. *Language Resources and Evaluation*, 47(4):919–944, 2013.

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10428–10436, 2020.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In Platt, J., Koller, D., Singer, Y., and Roweis, S. (eds.), *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2008. URL https://proceedings.neurips.cc/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf.

Ren, H., Dai, H., Dai, Z., Yang, M., Leskovec, J., Schuurmans, D., and Dai, B. Combiner: Full attention transformer with sparse computation cost. *Advances in Neural Information Processing Systems*, 34, 2021.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pp. 1278–1286, 2014.

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., and Fergus, R. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15), 2021. ISSN 0027-8424. doi: 10.1073/pnas.2016239118. URL https://www.pnas.org/content/118/15/e2016239118.

Roy, A., Saffar, M., Vaswani, A., and Grangier, D. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. doi: 10.1162/tacl_a_00353. URL https://aclanthology.org/2021.tacl-1.4.

Schlag, I., Irie, K., and Schmidhuber, J. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pp. 9355–9366. PMLR, 2021.

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Tay, Y., Bahri, D., Yang, L., Metzler, D., and Juan, D.-C. Sparse Sinkhorn attention. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9438–9447. PMLR, 13–18 Jul 2020a. URL https://proceedings.mlr.press/v119/tay20a.html.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020b.

Tay, Y., Bahri, D., Metzler, D., Juan, D.-C., Zhao, Z., and Zheng, C. Synthesizer: Rethinking self-attention for transformer models. In *International Conference on Machine Learning*, pp. 10183–10192. PMLR, 2021a.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=qVyeW-grC2k.

Titsias, M. and Lázaro-Gredilla, M. Doubly stochastic variational bayes for non-conjugate inference. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pp. 1971–1979, 2014.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/touvron21a.html.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Veach, E. and Guibas, L. J. Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 419–428, 1995.

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

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

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P., and Shao, L. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021a.

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P., and Shao, L. Pvtv2: Improved baselines with pyramid vision transformer. *arXiv preprint arXiv:2106.13797*, 2021b.

Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., and Zhang, L. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021.

Xiao, T., Singh, M., Mintun, E., Darrell, T., Dollár, P., and Girshick, R. Early convolutions help transformers see better. *arXiv preprint arXiv:2106.14881*, 2021.

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

Yang, J., Li, C., Zhang, P., Dai, X., Xiao, B., Yuan, L., and Gao, J. Focal attention for long-range interactions in vision transformers. *Advances in Neural Information Processing Systems*, 34, 2021.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023–6032, 2019.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. Big bird: Transformers for longer sequences. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17283–17297. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Zhang, P., Dai, X., Yang, J., Xiao, B., Yuan, L., Zhang, L., and Gao, J. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. *arXiv preprint arXiv:2103.15358*, 2021.

Zheng, L., Pan, H., and Kong, L. Ripple attention for visual perception with sub-quadratic complexity. *arXiv preprint arXiv:2110.02453*, 2021.

Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 13001–13008, 2020.

Zhu, C., Ping, W., Xiao, C., Shoeybi, M., Goldstein, T., Anandkumar, A., and Catanzaro, B. Long-short transformer: Efficient transformers for language and vision. *Advances in Neural Information Processing Systems*, 34, 2021.

Appendices

A. Proof for Proposition 3.1

Assume $q(\omega) = \mathcal{N}(\omega; 0, \mathbf{I})$. Recall that we define

$$\frac{p(\omega)}{q(\omega)} f(\omega) = \frac{1}{Z} \sum_{m=1}^{M} \xi(\mathbf{q}_n, \omega)^\top \xi(\mathbf{k}_m, \omega) \mathbf{v}_m^\top, \tag{11}$$

$$\frac{p(\omega)}{q(\omega)} = \frac{1}{Z} \sum_{m=1}^{M} \xi(\mathbf{q}_n, \omega)^\top \xi(\mathbf{k}_m, \omega). \tag{12}$$

On solving $f(\omega)$. Substituting the second equality into the first one yields the form of $f(\omega)$:

$$f(\omega) = \frac{\sum_{m=1}^{M} \xi(\mathbf{q}_n, \omega)^\top \xi(\mathbf{k}_m, \omega) \mathbf{v}_m^\top}{\sum_{m'=1}^{M} \xi(\mathbf{q}_n, \omega)^\top \xi(\mathbf{k}_{m'}, \omega)} = \frac{\xi(\mathbf{q}_n, \omega)^\top \sum_{m=1}^{M} \xi(\mathbf{k}_m, \omega) \mathbf{v}_m^\top}{\xi(\mathbf{q}_n, \omega)^\top \sum_{m'=1}^{M} \xi(\mathbf{k}_{m'}, \omega)}. \tag{13}$$

To more clearly illustrate the connection between RFA and RA, one can also manually verify this by rearranging terms in Equation 2:

$$\begin{aligned}
&\mathsf{RFA}\left(\mathbf{q}_n, \mathbf{K}, \mathbf{V}\right) \\
&= \frac{\sum_{s=1}^{S} \sum_{m=1}^{M} \xi(\mathbf{q}_n, \omega_s)^\top \xi(\mathbf{k}_m, \omega_s) \mathbf{v}_m^\top}{\sum_{s=1}^{S} \sum_{m'=1}^{M} \xi(\mathbf{q}_n, \omega_s)^\top \xi(\mathbf{k}_{m'}, \omega_s)} \\
&= \frac{\sum_{s=1}^{S} \left(\sum_{m'=1}^{M} \xi(\mathbf{q}_n, \omega_s)^\top \xi(\mathbf{k}_{m'}, \omega_s) \right) \frac{\sum_{m=1}^{M} \xi(\mathbf{q}_n, \omega_s)^\top \xi(\mathbf{k}_m, \omega_s) \mathbf{v}_m^\top}{\sum_{m'=1}^{M} \xi(\mathbf{q}_n, \omega_s)^\top \xi(\mathbf{k}_{m'}, \omega_s)}}{\sum_{s=1}^{S} \sum_{m'=1}^{M} \xi(\mathbf{q}_n, \omega_s)^\top \xi(\mathbf{k}_{m'}, \omega_s)} \\
&= \frac{\sum_{s=1}^{S} \frac{Zp(\omega_s)}{q(\omega_s)} \frac{\sum_{m=1}^{M} \xi(\mathbf{q}_n, \omega_s)^\top \xi(\mathbf{k}_m, \omega_s) \mathbf{v}_m^\top}{\sum_{m'=1}^{M} \xi(\mathbf{q}_n, \omega_s)^\top \xi(\mathbf{k}_{m'}, \omega_s)}}{\sum_{s=1}^{S} \frac{Zp(\omega_s)}{q(\omega_s)}} := \frac{\sum_{s=1}^{S} \frac{Zp(\omega_s)}{q(\omega_s)} f(\omega_s)}{\sum_{s=1}^{S} \frac{Zp(\omega_s)}{q(\omega_s)}}.
\end{aligned}$$

On solving $p(\omega)$. According to Equation 12, we have

$$p(\omega) = \frac{q(\omega) \left[\sum_{m=1}^{M} \xi(\mathbf{q}_n, \omega)^\top \xi(\mathbf{k}_m, \omega) \right]}{Z},$$

where Z is the partition function. Recall that in Equation 1

$$\mathbb{E}_{\omega \sim \mathcal{N}(\omega; 0, \mathbf{I})} \left[\xi(\mathbf{x}, \omega)^\top \xi(\mathbf{y}, \omega) \right] = \int \xi(\mathbf{x}, \omega)^\top \xi(\mathbf{y}, \omega) q(\omega) d\omega = \exp(\mathbf{x}^\top \mathbf{y}), \tag{14}$$

which further implies

$$Z = \int q(\omega) \left[\sum_{m=1}^{M} \xi(\mathbf{q}_n, \omega)^\top \xi(\mathbf{k}_m, \omega) \right] d\omega = \sum_{m=1}^{M} \int \xi(\mathbf{q}_n, \omega)^\top \xi(\mathbf{k}_m, \omega) q(\omega) d\omega = \sum_{m=1}^{M} \exp(\mathbf{q}_n^\top \mathbf{k}_m).$$

Therefore,

$$\begin{aligned}
p(\omega) &= q(\omega) \frac{\sum_{m=1}^{M} \xi(\mathbf{q}_n, \omega)^\top \xi(\mathbf{k}_m, \omega)}{\sum_{m'=1}^{M} \exp(\mathbf{q}_n^\top \mathbf{k}_{m'})} \qquad (15) \\
&= \sum_{m=1}^{M} \frac{\exp(\mathbf{q}_n^\top \mathbf{k}_m)}{\sum_{m'=1}^{M} \exp(\mathbf{q}_n^\top \mathbf{k}_{m'})} \frac{q(\omega) \xi(\mathbf{q}_n, \omega)^\top \xi(\mathbf{k}_m, \omega)}{\exp(\mathbf{q}_n^\top \mathbf{k}_m)} \\
&= \sum_{m=1}^{M} p(m) p(\omega|m), \qquad (16)
\end{aligned}$$

which is effectively a mixture distribution and each component is selected with probability proportional to the similarity of queries and keys. As long as the randomized mapping is non-negative, $p(\omega|m)$ would be a valid probability distribution since its density would be non-negative and integrate to 1, according to Equation 14.

In terms of the particular form of the distribution, we have the following lemma:

Lemma A.1. *Assume $\xi(\mathbf{x},\omega) = \exp\left(\omega^\top \mathbf{x} - \frac{\|\mathbf{x}\|^2}{2}\right)$ and $q(\omega) = \mathcal{N}(\omega; 0, \mathbf{I})$. Given two vectors $\mathbf{q}_n$ and $\mathbf{k}_m$ with the same dimension as $\omega \in \mathbb{R}^D$, if a density function $g(\omega)$ w.r.t. the random vector ω is defined as*

$$g(\omega) := \frac{q(\omega)\xi(\mathbf{q}_n,\omega)^\top \xi(\mathbf{k}_m,\omega)}{\exp(\mathbf{q}_n^\top \mathbf{k}_m)},$$

Then $\omega \sim \mathcal{N}(\omega; \mathbf{q}_n + \mathbf{k}_m, \mathbf{I})$.

Proof. Note that $q(\omega) = \mathcal{N}(\omega; 0, \mathbf{I}) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\omega^\top \omega\right)$. Based on the "complete the square" technique, we have

$$\begin{aligned}
g(\omega) &= \frac{q(\omega)\xi(\mathbf{q}_n,\omega)^\top \xi(\mathbf{k}_m,\omega)}{\exp(\mathbf{q}_n^\top \mathbf{k}_m)} \\
&= \frac{\exp\left(-\frac{1}{2}\omega^\top\omega\right)\exp\left(\omega^\top \mathbf{q}_n - \frac{\|\mathbf{q}_n\|^2}{2}\right)\exp\left(\omega^\top \mathbf{k}_m - \frac{\|\mathbf{k}_m\|^2}{2}\right)}{(2\pi)^{d/2}\exp(\mathbf{q}_n^\top \mathbf{k}_m)} \\
&= \frac{\exp\left(-\frac{1}{2}\omega^\top\omega + \omega^\top(\mathbf{q}_n + \mathbf{k}_m)\right)\exp\left(-\frac{1}{2}\|\mathbf{k}_m\|^2 - \frac{1}{2}\|\mathbf{q}_n\|^2\right)}{(2\pi)^{d/2}\exp(\mathbf{q}_n^\top \mathbf{k}_m)} \\
&= \frac{\exp\left(-\frac{1}{2}\omega^\top\omega + \omega^\top(\mathbf{q}_n + \mathbf{k}_m) - \frac{1}{2}(\mathbf{q}_n + \mathbf{k}_m)^\top(\mathbf{q}_n + \mathbf{k}_m)\right)\exp\left(\mathbf{q}_n^\top \mathbf{k}_m\right)}{(2\pi)^{d/2}\exp(\mathbf{q}_n^\top \mathbf{k}_m)} \\
&= \frac{1}{(2\pi)^{d/2}}\exp\left(-\frac{1}{2}\left[\omega - (\mathbf{q}_n + \mathbf{k}_m)\right]^\top\left[\omega - (\mathbf{q}_n + \mathbf{k}_m)\right]\right),
\end{aligned}$$

which is exactly the density function of a multivariate Gaussian with the mean $\mathbf{q}_n + \mathbf{k}_m$ and covariance $\mathbf{I}$. □

Following Lemma A.1, it is straightforward to obtain

$$\begin{aligned}
p(\omega) &= q(\omega)\frac{\sum_{m=1}^{M}\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)}{\sum_{m'=1}^{M}\exp(\mathbf{q}_n^\top\mathbf{k}_{m'})} \\
&= \sum_{m=1}^{M}\frac{q(\omega)\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)}{\sum_{m'=1}^{M}\exp(\mathbf{q}_n^\top\mathbf{k}_{m'})} \\
&= \sum_{m=1}^{M}\frac{\exp(\mathbf{q}_n^\top\mathbf{k}_m)}{\sum_{m'=1}^{M}\exp(\mathbf{q}_n^\top\mathbf{k}_{m'})}\frac{q(\omega)\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)}{\exp(\mathbf{q}_n^\top\mathbf{k}_m)} \\
&= \sum_{m=1}^{M}\frac{\exp(\mathbf{q}_n^\top\mathbf{k}_m)}{\sum_{m'=1}^{M}\exp(\mathbf{q}_n^\top\mathbf{k}_{m'})}\mathcal{N}(\omega;\mathbf{q}_n+\mathbf{k}_m,\mathbf{I}) \\
&:= \sum_{m=1}^{M}\pi_m\mathcal{N}(\omega;\mathbf{q}_n+\mathbf{k}_m,\mathbf{I}).
\end{aligned}$$

where $\pi_m = \frac{\exp(\mathbf{q}_n^\top\mathbf{k}_m)}{\sum_{m'=1}^{M}\exp(\mathbf{q}_n^\top\mathbf{k}_{m'})}$.

Discussion. Due to the dependence on the randomized mapping $\xi(\cdot,\cdot)$, different choices of feature maps would yield distinct density forms. Here we mainly study the positive randomized mapping in Performer (Choromanski et al., 2021) and leave other choices (such as trigonometric functions in Peng et al. (2021b)) as future work.

B. Proof for Proposition 3.2

Since the vanilla random-feature-based attention estimation is consistent, softmax attention must be equal to expected randomized attention. However, such equality can also be verified as follows. Assume $q(\omega) = \mathcal{N}(\omega; 0, \mathbf{I})$ and $p(\omega) = q(\omega)\frac{\sum_{m=1}^{M}\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)}{\sum_{m'=1}^{M}\exp(\mathbf{q}_n^\top\mathbf{k}_{m'})}$ given by Proposition 1. Then we have

$$\frac{q(\omega)}{p(\omega)} = \frac{\sum_{m'=1}^{M}\exp(\mathbf{k}_{m'}^\top\mathbf{q}_n)}{\sum_{m=1}^{M}\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)}. \tag{17}$$

In addition, according to the definition of randomized mappings $\xi(\cdot,\cdot)$,

$$\mathbb{E}_{\omega\sim\mathcal{N}(\omega;0,\mathbf{I})}\left[\xi(\mathbf{x},\omega)^\top\xi(\mathbf{y},\omega)\right] = \int \xi(\mathbf{x},\omega)^\top\xi(\mathbf{y},\omega)q(\omega)d\omega = \exp(\mathbf{x}^\top\mathbf{y}). \tag{18}$$

Equipped with these helpers, we are ready to derive the equality as follows:

$$\begin{aligned}
&\mathbb{E}\left[\mathsf{RA}(\mathbf{q}_n,\mathbf{K},\mathbf{V})\right]\\
&= \mathbb{E}_{p(\omega)}\left[\frac{\sum_{m=1}^{M}\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)\mathbf{v}_m^\top}{\sum_{m'=1}^{M}\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_{m'},\omega)}\right]\\
&= \mathbb{E}_{p(\omega)}\left[\frac{\sum_{m'=1}^{M}\exp(\mathbf{k}_{m'}^\top\mathbf{q}_n)}{\sum_{m=1}^{M}\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)}\frac{\sum_{m=1}^{M}\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)\mathbf{v}_m^\top}{\sum_{m'=1}^{M}\exp(\mathbf{k}_{m'}^\top\mathbf{q}_n)}\right]\\
&= \mathbb{E}_{p(\omega)}\left[\frac{q(\omega)}{p(\omega)}\frac{\sum_{m=1}^{M}\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)\mathbf{v}_m^\top}{\sum_{m'=1}^{M}\exp(\mathbf{k}_{m'}^\top\mathbf{q}_n)}\right] && \triangleright \text{ Equation 17}\\
&= \mathbb{E}_{q(\omega)}\left[\frac{\sum_{m=1}^{M}\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)\mathbf{v}_m^\top}{\sum_{m'=1}^{M}\exp(\mathbf{k}_{m'}^\top\mathbf{q}_n)}\right]\\
&= \frac{\sum_{m=1}^{M}\mathbb{E}_{q(\omega)}\left[\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)\right]\mathbf{v}_m^\top}{\sum_{m'=1}^{M}\exp(\mathbf{k}_{m'}^\top\mathbf{q}_n)} && \triangleright \text{ linearity of expectations}\\
&= \frac{\sum_{m=1}^{M}\exp(\mathbf{k}_m^\top\mathbf{q}_n)\mathbf{v}_m^\top}{\sum_{m'=1}^{M}\exp(\mathbf{k}_{m'}^\top\mathbf{q}_n)} && \triangleright \text{ Equation 18}\\
&= \mathsf{SoftmaxAttn}(\mathbf{q}_n,\mathbf{K},\mathbf{V})
\end{aligned}$$

C. Discussion on Different Randomized Mappings

The randomized mapping $\xi(\cdot,\cdot)$ transforms the inputs to a l-dimensional vector. There are various choices of $\xi(\cdot,\cdot)$ for the resulting estimator to become unbiased in the context of attention mechanisms, such as

- $l = 1$ and $\xi(\mathbf{x},\omega) = \exp\left(\omega^\top\mathbf{x} - \frac{\|\mathbf{x}\|^2}{2}\right)$ in Choromanski et al. (2021);
- $l = 1$ and $\xi(\mathbf{x},\omega) = \sqrt{2}\exp\left(\frac{\|\mathbf{x}\|^2}{2}\right)\cos\left(\omega^\top\mathbf{x} + b\right)$ with $b \sim \mathrm{Uniform}(0, 2\pi)$ in Rahimi & Recht (2008);
- $l = 2$ and $\xi(\mathbf{x},\omega) = \left[\exp\left(\frac{\|\mathbf{x}\|^2}{2}\right)\sin\left(\omega^\top\mathbf{x}\right), \exp\left(\frac{\|\mathbf{x}\|^2}{2}\right)\cos\left(\omega^\top\mathbf{x}\right)\right]$ in Rahimi & Recht (2008); Peng et al. (2021b);
- $l = 2$ and $\xi(\mathbf{x},\omega) = \left[\frac{1}{\sqrt{2}}\exp\left(\omega^\top\mathbf{x} - \frac{\|\mathbf{x}\|^2}{2}\right), \frac{1}{\sqrt{2}}\exp\left(-\omega^\top\mathbf{x} - \frac{\|\mathbf{x}\|^2}{2}\right)\right]$ in Choromanski et al. (2021).

In the main paper, we focus on the positive randomized mappings (Choromanski et al., 2021); for other positive randomized mappings, it is also possible to derive a similar target expectation, such as the hyperbolic randomized mapping proposed in Choromanski et al. (2021):

Corollary C.1. *Consider the hyperbolic randomized mapping*

$$\xi(\mathbf{x},\omega)=\frac{1}{\sqrt{2}}\exp\left(-\frac{\|\mathbf{x}\|^2}{2}\right)\left[\exp\left(\omega^\top\mathbf{x}\right),\exp\left(-\omega^\top\mathbf{x}\right)\right]^\top.$$

It also implies an SNIS estimator of $\mathbb{E}_{p(\omega)}\left[f(\omega)\right]$, where the function $f(\omega)$ remains the same as Equation 6 and the density $p(\omega)$ is also a Gaussian mixture as follows:

$$\frac{1}{2}\sum_{m=1}^{M}\pi_m\left(\mathcal{N}(\omega;\mathbf{q}_n+\mathbf{k}_m,\mathbf{I})+\mathcal{N}(\omega;-\mathbf{q}_n-\mathbf{k}_m,\mathbf{I})\right).$$

Proof. Consider the hyperbolic positive randomized mapping

$$\xi(\mathbf{x},\omega)=\frac{1}{\sqrt{2}}\exp\left(-\frac{\|\mathbf{x}\|^2}{2}\right)\left[\exp\left(\omega^\top\mathbf{x}\right),\exp\left(-\omega^\top\mathbf{x}\right)\right]^\top.$$

According to proof of Proposition 3.1 in Appendix A, the density function $p(\omega)$ corresponding to the hyperbolic randomized mapping should also be a mixture (Equation 16) with the following form

$$p(\omega)=\sum_{m=1}^{M}\frac{\exp(\mathbf{q}_n^\top\mathbf{k}_m)}{\sum_{m'=1}^{M}\exp(\mathbf{q}_n^\top\mathbf{k}_{m'})}\frac{q(\omega)\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)}{\exp(\mathbf{q}_n^\top\mathbf{k}_m)}:=\sum_{m=1}^{M}\pi_m p(\omega|m),$$

where $\pi_m:=\frac{\exp(\mathbf{q}_n^\top\mathbf{k}_m)}{\sum_{m'=1}^{M}\exp(\mathbf{q}_n^\top\mathbf{k}_{m'})}$ and $p(\omega|m)$ denotes the density of the m-th component distribution. By substituting the hyperbolic randomized mapping into the equation above, we have

$$\begin{aligned}
&p(\omega|m)\\
&=\frac{q(\omega)\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)}{\exp(\mathbf{q}_n^\top\mathbf{k}_m)}\\
&=\frac{1}{2}\frac{q(\omega)\left[\exp\left(\omega^\top\mathbf{q}_n-\frac{\|\mathbf{q}_n\|^2}{2}\right)\exp\left(\omega^\top\mathbf{k}_m-\frac{\|\mathbf{k}_m\|^2}{2}\right)+\exp\left(-\omega^\top\mathbf{q}_n-\frac{\|\mathbf{q}_n\|^2}{2}\right)\exp\left(-\omega^\top\mathbf{k}_m-\frac{\|\mathbf{k}_m\|^2}{2}\right)\right]}{\exp(\mathbf{q}_n^\top\mathbf{k}_m)}\\
&=\frac{1}{2}\frac{q(\omega)\exp\left(\omega^\top\mathbf{q}_n-\frac{\|\mathbf{q}_n\|^2}{2}\right)\exp\left(\omega^\top\mathbf{k}_m-\frac{\|\mathbf{k}_m\|^2}{2}\right)}{\exp(\mathbf{q}_n^\top\mathbf{k}_m)}+\frac{1}{2}\frac{q(\omega)\exp\left(-\omega^\top\mathbf{q}_n-\frac{\|\mathbf{q}_n\|^2}{2}\right)\exp\left(-\omega^\top\mathbf{k}_m-\frac{\|\mathbf{k}_m\|^2}{2}\right)}{\exp(\mathbf{q}_n^\top\mathbf{k}_m)}
\end{aligned}$$

It is straightforward to recognize that this can be viewed as the sum of two densities. We then invoke Lemma A.1 for each of them, which results in two Gaussians

$$p(\omega|m)=\frac{1}{2}\mathcal{N}(\omega;\mathbf{q}_n+\mathbf{k}_m,\mathbf{I})+\frac{1}{2}\mathcal{N}(\omega;-\mathbf{q}_n-\mathbf{k}_m,\mathbf{I}).$$

Therefore, the true density function $p(\omega)$ can be expressed as follows

$$p(\omega)=\pi_m p(\omega|m)=\frac{1}{2}\sum_{m=1}^{M}\pi_m\left(\mathcal{N}(\omega;\mathbf{q}_n+\mathbf{k}_m,\mathbf{I})+\mathcal{N}(\omega;-\mathbf{q}_n-\mathbf{k}_m,\mathbf{I})\right).$$

□

However, it is much more difficult to analyze the classical random Fourier mappings (Rahimi & Recht, 2008) since they may involve a negative density. As a result, the formulation of RFA with these randomized mappings may not define a valid self-normalized importance sampling estimate. We study positive randomized mappings through this paper and leave investigation into other cases as future work.

D. Analysis on the Optimal Weighting Function in Multiple Importance Sampling

In this section, we analyze the optimal weighting function in MIS, which is self-normalized in our setting (§4.2).

Given the set of N queries $\mathbf{Q}$ and the set of M key-value pairs $\mathbf{K}$ and $\mathbf{V}$, the regular softmax attention can be expressed as expected randomized attention according to Equation 7:

$$\frac{\sum_{m=1}^{M}\exp(\mathbf{q}_n^\top\mathbf{k}_m)\mathbf{v}_m^\top}{\sum_{m'=1}^{M}\exp(\mathbf{q}_n^\top\mathbf{k}_{m'})} = \mathbb{E}_{p_n(\omega)}\left[\frac{\sum_{m=1}^{M}\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)\mathbf{v}_m^\top}{\sum_{m'=1}^{M}\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_{m'},\omega)}\right] := \mathbb{E}_{p_n(\omega)}\left[f_n(\omega)\right] = \boldsymbol{\mu}_n,$$

where the distribution is defined in Proposition 3.1 as

$$p_n(\omega) = \mathcal{N}(\omega;0,\mathbf{I})\frac{\sum_{m=1}^{M}\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)}{\sum_{m'=1}^{M}\exp(\mathbf{q}_n^\top\mathbf{k}_{m'})} = \sum_{m=1}^{M}\frac{\exp(\mathbf{q}_n^\top\mathbf{k}_m)}{\sum_{m'=1}^{M}\exp(\mathbf{q}_n^\top\mathbf{k}_{m'})}\mathcal{N}(\omega;\mathbf{q}_n+\mathbf{k}_m,\mathbf{I}).$$

The attention mechanism outputs a D-dimensional vector for each query. For brevity, we start with considering the d-th dimension and denote $f_{n,d}(\omega)$ as the d-th dimension of the function output at query position n. We then have

$$\mathbb{E}_{p_n(\omega)}\left[f_{n,d}(\omega)\right] = \mathbb{E}_{p_n(\omega)}\left[\frac{\sum_{m=1}^{M}\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_m,\omega)v_{m,d}}{\sum_{m'=1}^{M}\xi(\mathbf{q}_n,\omega)^\top\xi(\mathbf{k}_{m'},\omega)}\right] = \frac{\sum_{m=1}^{M}\exp(\mathbf{q}_n^\top\mathbf{k}_m)v_{m,d}}{\sum_{m'=1}^{M}\exp(\mathbf{q}_n^\top\mathbf{k}_{m'})} := \mu_{n,d}.$$

In our work, we estimate the expectation above by self-normalized multiple importance sampling (see §4.2). For the d-th dimension of the output at query position n, we have

$$\hat{g}_{n,d} := \frac{\sum_{c=1}^{C}\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}f_{n,d}(\omega)\right]}{\sum_{c=1}^{C}\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}\right]} \approx \frac{\sum_{c=1}^{C}\alpha_{nc}(\omega_c)\frac{p_n(\omega_c)}{q_c(\omega_c)}f_{n,d}(\omega_c)}{\sum_{c=1}^{C}\alpha_{nc}(\omega_c)\frac{p_n(\omega_c)}{q_c(\omega_c)}} := \frac{A}{B},$$

where $\omega_c \sim p_c(\omega)$ for $c = 1,\ldots,C$. We also let A and B represent the nominator and denominator respectively. The expectations of A and B are

$$\mu_A := \sum_{c=1}^{C}\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}f_{n,d}(\omega)\right] = \mathbb{E}_{p_n(\omega)}\left[f_{n,d}(\omega)\right] = \mu_{n,d};$$
$$\mu_B := \sum_{c=1}^{C}\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}\right] = \mathbb{E}_{p_n(\omega)}\left[1\right] = 1.$$

Unfortunately, the exact form of the variance of $\hat{g}_{n,d}$ is mostly intractable to compute. To this end, we follow previous practices (Owen, 2013) and approximate $\mathrm{Var}\left[\hat{g}_{n,d}\right]$ via the delta method. In particular, we apply the first-order Taylor expansion approximation to the function $g(A,B) := A/B$ around point (μ_A,μ_B), yielding

$$\begin{aligned}\frac{A}{B} = g(A,B) &\approx g(\mu_A,\mu_B) + \frac{\partial g(A,B)}{\partial A}\Big|_{\substack{A=\mu_A\\B=\mu_B}}(A-\mu_A) + \frac{\partial g(A,B)}{\partial B}\Big|_{\substack{A=\mu_A\\B=\mu_B}}(B-\mu_B)\\ &:= g(\mu_A,\mu_B) + g_A(A-\mu_A) + g_B(B-\mu_B),\end{aligned} \tag{19}$$

where we denote $g_A := \frac{\partial g(A,B)}{\partial A}\Big|_{\substack{A=\mu_A\\B=\mu_B}}$ and $g_B := \frac{\partial g(A,B)}{\partial B}\Big|_{\substack{A=\mu_A\\B=\mu_B}}$ similarly. Note that both g_A and g_B are constants with respect to ω. According to Equation 19, the approximate expectation is the following

$$\begin{aligned}\mathbb{E}\left[g(A,B)\right] &\approx \mathbb{E}\left[g(\mu_A,\mu_B)\right] + \mathbb{E}\left[g_A(A-\mu_A) + g_B(B-\mu_B)\right]\\ &= g(\mu_A,\mu_B) + g_A\mathbb{E}\left[(A-\mu_A)\right] + g_B\mathbb{E}\left[(B-\mu_B)\right]\\ &= g(\mu_A,\mu_B)\end{aligned}$$

and its second moment is given by

$$\begin{aligned}&\mathbb{E}\left[g(A,B)^2\right]\\ &= \mathbb{E}\left[g(\mu_A,\mu_B)^2 + g_A^2(A-\mu_A)^2 + g_B^2(B-\mu_B)^2 + 2g_Ag_B(A-\mu_A)(B-\mu_B)\right]\\ &= g(\mu_A,\mu_B)^2 + g_A^2\,\mathrm{Var}\left[A\right] + g_B^2\,\mathrm{Var}\left[B\right] + 2g_Ag_B\,\mathrm{Cov}\left(A,B\right).\end{aligned}$$

It is straightforward to compute that

$$\begin{aligned}
\operatorname{Var}\left[A\right] &= \sum_{c=1}^{C} \operatorname{Var}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}f_{n,d}(\omega)\right] \\
&= \sum_{c=1}^{C} \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}^2(\omega)\frac{p_n^2(\omega)}{q_c^2(\omega)}f_{n,d}^2(\omega)\right] - \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}f_{n,d}(\omega)\right]^2 \\
\operatorname{Var}\left[B\right] &= \sum_{c=1}^{C} \operatorname{Var}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}\right] = \sum_{c=1}^{C} \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}^2(\omega)\frac{p_n^2(\omega)}{q_c^2(\omega)}\right] - \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}\right]^2 \\
\operatorname{Cov}\left(A,B\right) &= \sum_{c=1}^{C}\sum_{c'=1}^{C} \operatorname{Cov}\left(\alpha_{nc}(\omega_c)\frac{p_n(\omega_c)}{q_c(\omega_c)}f_{n,d}(\omega_c), \alpha_{nc'}(\omega_{c'})\frac{p_n(\omega_{c'})}{q_{c'}(\omega_{c'})}\right) \\
&= \sum_{c=1}^{C} \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}^2(\omega)\frac{p_n^2(\omega)}{q_c^2(\omega)}f_{n,d}(\omega)\right] - \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}f_{n,d}(\omega)\right]\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}\right]. \\
g_A^2 &= \frac{1}{\mu_B^2} = 1, \\
g_B^2 &= \frac{\mu_A^2}{\mu_B^4} = \mu_{n,d}^2, \\
g_A g_B &= -\frac{\mu_A}{\mu_B^3} = -\mu_{n,d}.
\end{aligned}$$

The first three lines hold since ω_c is independent of $\omega_{c'}$ for any $c \neq c'$. Therefore, the approximate variance of our estimate at the d-th dimension can be written as

$$\begin{aligned}
&\operatorname{Var}\left[\hat{g}_{n,d}\right] \\
&= \operatorname{Var}\left[g(A,B)\right] \\
&= \mathbb{E}\left[g(A,B)^2\right] - \mathbb{E}\left[g(A,B)\right]^2 \\
&\approx g_A^2 \operatorname{Var}\left[A\right] + g_B^2 \operatorname{Var}\left[B\right] + 2g_A g_B \operatorname{Cov}\left(A,B\right) \\
&= \sum_{c=1}^{C}\left(\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}^2(\omega)\frac{p_n^2(\omega)}{q_c^2(\omega)}f_{n,d}^2(\omega)\right] - \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}f_{n,d}(\omega)\right]^2\right) + \\
&\qquad \mu_{n,d}^2\left(\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}^2(\omega)\frac{p_n^2(\omega)}{q_c^2(\omega)}\right] - \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}\right]^2\right) - \\
&\qquad 2\mu_{n,d}\left(\sum_{c=1}^{C}\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}^2(\omega)\frac{p_n^2(\omega)}{q_c^2(\omega)}f_{n,d}(\omega)\right] - \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}f_{n,d}(\omega)\right]\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}\right]\right) \\
&= \sum_{c=1}^{C}\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}^2(\omega)\frac{p_n^2(\omega)}{q_c^2(\omega)}\left(f_{n,d}^2(\omega) - 2f_{n,d}(\omega)\mu_{n,d} + \mu_{n,d}^2\right)\right] - \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}f_{n,d}(\omega)\right]^2 - \\
&\qquad \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}\mu_{n,d}\right]^2 + 2\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}f_{n,d}(\omega)\right]\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}\mu_{n,d}\right] \\
&= \sum_{c=1}^{C}\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}^2(\omega)\frac{p_n^2(\omega)}{q_c^2(\omega)}\left(f_{n,d}(\omega) - \mu_{n,d}\right)^2\right] - \left(\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}f_{n,d}(\omega)\right] - \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}\mu_{n,d}\right]\right)^2 \\
&= \sum_{c=1}^{C}\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}^2(\omega)\frac{p_n^2(\omega)}{q_c^2(\omega)}\left(f_{n,d}(\omega) - \mu_{n,d}\right)^2\right] - \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}\left(f_{n,d}(\omega) - \mu_{n,d}\right)\right]^2.
\end{aligned}$$

Since we are using the same proposal distribution to estimate the output for all dimensions, we are interested in the sum of variance over every dimension (i.e., the trace of the covariance matrix):

$$\begin{aligned}\sum_{d=1}^{D} \mathrm{Var}\left[\hat{g}_{n,d}\right] &\approx \sum_{d=1}^{D}\sum_{c=1}^{C} \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}^2(\omega)\frac{p_n^2(\omega)}{q_c^2(\omega)}\left(f_{n,d}(\omega)-\mu_{n,d}\right)^2\right] - \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}\left(f_{n,d}(\omega)-\mu_{n,d}\right)\right]^2 \\ &= \sum_{c=1}^{C} \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}^2(\omega)\frac{p_n^2(\omega)}{q_c^2(\omega)}\sum_{d=1}^{D}\left(f_{n,d}(\omega)-\mu_{n,d}\right)^2\right] - \sum_{d=1}^{D}\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}\left(f_{n,d}(\omega)-\mu_{n,d}\right)\right]^2 \\ &= \sum_{c=1}^{C} \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}^2(\omega)\frac{p_n^2(\omega)}{q_c^2(\omega)}\|f_n(\omega)-\boldsymbol{\mu}_n\|^2\right] - \sum_{d=1}^{D}\mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega)\frac{p_n(\omega)}{q_c(\omega)}\left(f_{n,d}(\omega)-\mu_{n,d}\right)\right]^2. \end{aligned} \tag{20}$$

According to our design choice, $\alpha_{nc}(\cdot)$ is specific to each query position. Ideally, we hope these weighting functions can minimize the sum of variance at each position. Formally, we have

$$\begin{aligned} \underset{\{\alpha_{nc}\}_{c=1}^{C}}{\text{minimize}} \quad & \sum_{d=1}^{D} \mathrm{Var}\left[\hat{g}_{n,d}\right] \\ \text{subject to} \quad & \sum_{c=1}^{C}\alpha_{nc}(\omega) = 1 \text{ for any } \omega. \end{aligned} \tag{21}$$

Optimizing weighting functions to minimize the variance of ordinary MIS estimator has been studied by a recent work (Kondapaneni et al., 2019). Our setting is different from it in that (1) we focus on self-normalized MIS and that (2) the function $f(\cdot)$ is vector-valued instead of scalar-valued. These differences lead to a distinct objective (Equation 20). Here we adapt the analysis (Kondapaneni et al., 2019) to solve this problem. In particular, we rely on the calculus of variations and introduce the following Lagrangian

$$\mathcal{L}(\alpha,\lambda) = \sum_{d=1}^{D}\mathrm{Var}\left[\hat{g}_{n,d}\right] - \int \lambda\left(\sum_{c=1}^{C}\alpha_{nc}(\omega) - 1\right)d\omega. \tag{22}$$

Solving $\frac{\partial\mathcal{L}(\alpha,\lambda)}{\partial\alpha_{nc}} = 0$ and $\frac{\partial\mathcal{L}(\alpha,\lambda)}{\partial\lambda} = 0$ respectively yields

$$2\alpha_{nc}(\omega)\frac{p_n^2(\omega)}{q_c(\omega)}\|f_n(\omega)-\boldsymbol{\mu}_n\|^2 - 2\sum_{d=1}^{D}p_n(\omega)\left(f_{n,d}(\omega)-\mu_{n,d}\right)r_{ncd} - \lambda = 0; \tag{23}$$

$$\sum_{c=1}^{C}\alpha_{nc}(\omega) = 1. \tag{24}$$

Here we denote $r_{ncd} \coloneqq \int \alpha_{nc}(\omega)p_n(\omega)\left(f_{n,d}(\omega)-\mu_{n,d}\right)d\omega$. We then rearrange Equation 23 to obtain

$$\alpha_{nc}(\omega) = \frac{q_c(\omega)}{2p_n^2(\omega)\|f_n(\omega)-\boldsymbol{\mu}_n\|^2}\lambda + q_c(\omega)\frac{\sum_{d=1}^{D}p_n(\omega)\left(f_{n,d}(\omega)-\mu_{n,d}\right)r_{ncd}}{p_n^2(\omega)\|f_n(\omega)-\boldsymbol{\mu}_n\|^2}. \tag{25}$$

Substituting Equation 25 into Equation 24 gives

$$\lambda = \frac{2p_n^2(\omega)\|f_n(\omega)-\boldsymbol{\mu}_n\|^2}{\sum_{c=1}^{C}q_c(\omega)} - 2\frac{\sum_{c=1}^{C}q_c(\omega)\sum_{d=1}^{D}p_n(\omega)\left(f_{n,d}(\omega)-\mu_{n,d}\right)r_{ncd}}{\sum_{c'=1}^{C}q_{c'}(\omega)}. \tag{26}$$

Substituting Equation 26 back into Equation 25 yields

$$\begin{aligned}
&\alpha_{nc}(\omega) \\
&= \frac{q_c(\omega)}{\sum_{c=1}^C q_c(\omega)}\left(1 - \frac{\sum_{c=1}^C q_c(\omega)\sum_{d=1}^D p_n(\omega)\left(f_{n,d}(\omega) - \mu_{n,d}\right) r_{ncd}}{p_n^2(\omega)\|f_n(\omega) - \boldsymbol{\mu}_n\|^2}\right) + q_c(\omega)\frac{\sum_{d=1}^D p_n(\omega)\left(f_{n,d}(\omega) - \mu_{n,d}\right) r_{ncd}}{p_n^2(\omega)\|f_n(\omega) - \boldsymbol{\mu}_n\|^2} \\
&= \frac{q_c(\omega)}{\sum_{c=1}^C q_c(\omega)}\left(1 - \sum_{c=1}^C q_c(\omega) r_{nc}(\omega)\right) + q_c(\omega) r_{nc}(\omega) \\
&= \frac{q_c(\omega)}{\sum_{c'=1}^C q_{c'}(\omega)} + q_c(\omega)\left(r_{nc}(\omega) - \sum_{c=1}^C \frac{q_c(\omega)}{\sum_{c'=1}^C q_{c'}(\omega)} r_{nc}(\omega)\right)
\end{aligned} \tag{27}$$

where we denote

$$r_{nc}(\omega) := \frac{\sum_{d=1}^D p_n(\omega)\left(f_{n,d}(\omega) - \mu_{n,d}\right) r_{ncd}}{p_n^2(\omega)\|f_n(\omega) - \boldsymbol{\mu}_n\|^2}.$$

The characteristic of existence and uniqueness of the optimal weighting function is similar to Kondapaneni et al. (2019). Intuitively, the optimal weighting functions can be obtained by first calculating a query-dependent correction term, which sums to 0, and then adding such correction to the original balance heuristic weighting function. For large r_{nc}, the correction term will be positive, driving the weights for the c-th proposal to be higher; and vice versa. Such formulation introduces the dependence between the current proposal index c and the target query position n, which allows the weighting functions α_{nc} (and thus the estimator) to specialize in the current query.

To obtain the exact form of $r_{nc}(\omega)$, we need to solve $r_{ncd} = \int \alpha_{nc}(\omega) p_n(\omega)\left(f_{n,d}(\omega) - \mu_{n,d}\right) d\omega$. However, deriving a closed form solution is mostly intractable given its complex structure, which not only involves an intractable integral but also mixes together the effect from different dimensions. To further analyze this problem, we start with a simplified case where $D = 1$. In this setting, we have the following:

$$\begin{aligned}
&\sum_{c=1}^C r_{ncD}\int \frac{q_{c'}(\omega) q_c(\omega)}{\sum_{c=1}^C q_c(\omega)} d\omega = \int \frac{q_{c'}(\omega) p_n(\omega)(f_{n,D}(\omega) - \mu_{n,D})}{\sum_{c=1}^C q_c(\omega)} d\omega, \\
&r_{nc}(\omega) := \frac{p_n(\omega)\left(f_{n,D}(\omega) - \mu_{n,D}\right) r_{ncD}}{p_n^2(\omega)(f_{n,D}(\omega) - \mu_{n,D})^2}.
\end{aligned}$$

for any $c' = 1, \dots, C$. Although solving this linear system is intractable, it indicates that r_{ncD} roughly describes how $q_c(\omega)$ aligns with $p_n(\omega)(f_{n,d}(\omega) - \mu_{n,d})$ under the expectation of different q_c'. Therefore, $r_{nc}(\omega)$ can be seen as an indicator for the correlation between the current proposal $q_c(\omega)$ and $p_n(\omega)(f_{n,d}(\omega) - \mu_{n,d})$ that is normalized by the strength of $p_n(\omega)\left(f_{n,D}(\omega) - \mu_{n,D}\right)$.

For larger D, such concise equality involving r_{ncd} is not available since the effect of different dimensions is mixed. We thus seek an heuristic approximation that not only reflects the same intuition but also becomes tractable in practice (see Appendix G.3.2 for practical implementations).

E. Derivation for the Formulation of LARA

In this section, we give the detailed derivation for the final expression of our estimator LARA:

$$\begin{aligned}
&\mathsf{LARA}\left(\mathbf{q}_n, \mathbf{K}, \mathbf{V}\right) \\
&= \frac{\sum_{c=1}^C \alpha'_{nc}(\omega_c)\xi(\mathbf{q}_n, \omega_c)^\top \sum_{m=1}^M \xi(\mathbf{k}_m, \omega_c)\mathbf{v}_m^\top}{\sum_{c=1}^C \alpha'_{nc}(\omega_c)\xi(\mathbf{q}_n, \omega_c)^\top \sum_{m=1}^M \xi(\mathbf{k}_m, \omega_c)},
\end{aligned} \tag{28}$$

First, recall that

$$\begin{aligned}
p_n(\omega) &= \frac{\mathcal{N}(\omega; 0, \mathbf{I})\sum_{m=1}^M \xi(\mathbf{q}_n, \omega)^\top \xi(\mathbf{k}_m, \omega)}{\sum_{m'=1}^M \exp\left(\mathbf{q}_n^\top \mathbf{k}_{m'}\right)} := \frac{\tilde{p}_n(\omega)}{Z_p}; \\
f_n(\omega) &= \frac{\sum_{m=1}^M \xi(\mathbf{q}_n, \omega)^\top \xi(\mathbf{k}_m, \omega)\mathbf{v}_m^\top}{\sum_{m'=1}^M \xi(\mathbf{q}_n, \omega)^\top \xi(\mathbf{k}_{m'}, \omega)},
\end{aligned}$$

The formulation (Equation 28) is obtained by substituting the equations above into the self-normalized estimator:

$$\frac{\sum_{c=1}^{C} \alpha_{nc}(\omega_c) \frac{\tilde{p}_n(\omega_c)}{q_c(\omega_c)} f_n(\omega_c)}{\sum_{c=1}^{C} \alpha_{nc}(\omega_c) \frac{\tilde{p}_n(\omega_c)}{q_c(\omega_c)}} = \frac{\sum_{c=1}^{C} \alpha_{nc}(\omega_c) \frac{\mathcal{N}(\omega_c;0,\mathbf{I}) \sum_{m=1}^{M} \xi(\mathbf{q}_n,\omega_c)^\top \xi(\mathbf{k}_m,\omega_c)}{q_c(\omega_c)} \frac{\sum_{m=1}^{M} \xi(\mathbf{q}_n,\omega_c)^\top \xi(\mathbf{k}_m,\omega_c)\mathbf{v}_m^\top}{\sum_{m'=1}^{M} \xi(\mathbf{q}_n,\omega_c)^\top \xi(\mathbf{k}_{m'},\omega_c)}}{\sum_{c=1}^{C} \alpha_{nc}(\omega_c) \frac{\mathcal{N}(\omega_c;0,\mathbf{I}) \sum_{m=1}^{M} \xi(\mathbf{q}_n,\omega_c)^\top \xi(\mathbf{k}_m,\omega_c)}{q_c(\omega_c)}}$$

$$= \frac{\sum_{c=1}^{C} \alpha_{nc}(\omega_c) \frac{\mathcal{N}(\omega_c;0,\mathbf{I}) \sum_{m=1}^{M} \xi(\mathbf{q}_n,\omega_c)^\top \xi(\mathbf{k}_m,\omega_c)\mathbf{v}_m^\top}{q_c(\omega_c)}}{\sum_{c=1}^{C} \alpha_{nc}(\omega_c) \frac{\mathcal{N}(\omega_c;0,\mathbf{I}) \sum_{m=1}^{M} \xi(\mathbf{q}_n,\omega_c)^\top \xi(\mathbf{k}_m,\omega_c)}{q_c(\omega_c)}}$$

$$= \frac{\sum_{c=1}^{C} \alpha_{nc}(\omega_c) \frac{\mathcal{N}(\omega_c;0,\mathbf{I})}{q_c(\omega_c)} \sum_{m=1}^{M} \xi(\mathbf{q}_n,\omega_c)^\top \xi(\mathbf{k}_m,\omega_c)\mathbf{v}_m^\top}{\sum_{c=1}^{C} \alpha_{nc}(\omega_c) \frac{\mathcal{N}(\omega_c;0,\mathbf{I})}{q_c(\omega_c)} \sum_{m=1}^{M} \xi(\mathbf{q}_n,\omega_c)^\top \xi(\mathbf{k}_m,\omega_c)}$$

$$:= \frac{\sum_{c=1}^{C} \alpha'_{nc}(\omega_c) \sum_{m=1}^{M} \xi(\mathbf{q}_n,\omega_c)^\top \xi(\mathbf{k}_m,\omega_c)\mathbf{v}_m^\top}{\sum_{c=1}^{C} \alpha'_{nc}(\omega_c) \sum_{m=1}^{M} \xi(\mathbf{q}_n,\omega_c)^\top \xi(\mathbf{k}_{m'},\omega_c)} = \mathsf{LARA}\left(\mathbf{q}_n, \mathbf{K}, \mathbf{V}\right).$$

Note that we define $\alpha'_{nc}(\omega_c) := \alpha_{nc}(\omega_c) \frac{\mathcal{N}(\omega_c;0,\mathbf{I})}{q_c(\omega_c)}$.

F. Proof for the Unbiasedness of Multiple Importance Sampling

As in §4.1, suppose our MIS estimator takes the following form

$$\hat{g}_n = \sum_{c=1}^{C} \alpha_{nc}(\omega_c) \frac{p_n(\omega_c)}{q_c(\omega_c)} f_n(\omega_c), \quad \omega_c \sim q_c(\omega).$$

If $\sum_{c=1}^{C} \alpha_{nc}(\omega) = 1$, it can be shown that (Veach & Guibas, 1995)

$$\mathbb{E}\left[\hat{g}_n\right] = \sum_{c=1}^{C} \mathbb{E}_{q_c(\omega)}\left[\alpha_{nc}(\omega) \frac{p_n(\omega)}{q_c(\omega)} f_n(\omega)\right] = \sum_{c=1}^{C} \int \alpha_{nc}(\omega) p_n(\omega) f_n(\omega) d\omega$$
$$= \int \sum_{c=1}^{C} \alpha_{nc}(\omega) p_n(\omega) f_n(\omega) d\omega = \int p_n(\omega) f_n(\omega) d\omega = \mathbb{E}_{p_n(\omega)}\left[f_n(\omega)\right].$$

G. Details of RA, RFA and LARA

G.1. Sepcifics of Random Feature Attention

Some implementations of RFA (including Performer (Choromanski et al., 2021)) defines a sample-redrawing schedule, where the involved samples ω are periodically redrawn according to a hand-crafted strategy. However, this requires a task-specific specification and we found tuning redrawing strategies only brings marginal performance gain over the simplest method that redraws samples at each training iteration (we use the same sample set during the entire evaluation phase). Therefore, we adopt this method to train Performer for all tasks. We also do not use orthogonal random samples as in Choromanski et al. (2021), as we found it does not improve empirical performance but increases the training time. Algorithm 2 provides a algorithm sketch for random feature attention and linear randomized attention, respectively. Note that every loop involved in all provided pseudo-codes (Algorithm 1, Algorithm 2 and Algorithm 3) can be trivially executed in parallel.

G.2. Specifics of Randomized Attention

In this section, we describe the details of RA approximation for softmax attention. Recall in Proposition 3.1 the RA sampling distribution is a Gaussian mixture

$$p_n(\omega) = \sum_{m=1}^{M} \frac{\exp(\mathbf{q}_n^\top \mathbf{k}_m)}{\sum_{m'=1}^{M} \exp(\mathbf{q}_n^\top \mathbf{k}_{m'})} \mathcal{N}(\omega; \mathbf{q}_n + \mathbf{k}_m, \mathbf{I}) := \sum_{m=1}^{M} \pi_{nm} \mathcal{N}(\omega; \boldsymbol{\mu}_{nm}, \mathbf{I}) \tag{29}$$

with $\pi_{nm} = \frac{\exp(\mathbf{q}_n^\top \mathbf{k}_m)}{\sum_{m'=1}^{M} \exp(\mathbf{q}_n^\top \mathbf{k}_{m'})}$ and $\boldsymbol{\mu}_{nm} = \mathbf{q}_n + \mathbf{k}_m$. To sample from this Gaussian mixture distribution, we first sample $z_n \sim \mathrm{Categorical}(z; \boldsymbol{\pi}_n)$ with $\boldsymbol{\pi}_n$ being the probability masses at M possible outcomes and then let $\mathbf{a}_n := [a_{n1}, \ldots, a_{nM}]$

be an M-dimensional one-hot vector with $a_{nz_n} = 1$. The discrete random variable $\mathbf{a}_n$ defines which distribution component is selected. Since all components are Gaussian, we leverage reparameterization trick (Kingma & Welling, 2013; Rezende et al., 2014; Titsias & Lázaro-Gredilla, 2014) to draw independent $\epsilon \sim \mathcal{N}(\omega; 0, \mathbf{I})$ and add it to the selected mean, resulting in the final mixture sample. Formally, we express the sample ω_n from the Gaussian mixture as follows:

$$\omega_n = \sum_{m=1}^{M} a_{nm}\boldsymbol{\mu}_{nm} + \epsilon = \sum_{m=1}^{M} a_{nm}(\mathbf{q}_n + \mathbf{k}_m) + \epsilon = \mathbf{K}\mathbf{a}_n + \mathbf{q}_n + \epsilon, \quad \epsilon \sim \mathcal{N}(\epsilon; 0, \mathbf{I}), \tag{30}$$

which is then used to compute $f_n(\omega_n)$ to obtain the RA estimation (see Algorithm 1 for a algorithm sketch). Assuming the number of samples is S and the sequence length is N, the overall time/space complexity for RA is $\mathcal{O}(SN^2)$. Through experiments we take $S = 1$ sample in our randomized attention unless specified otherwise. We found this choice suffices to achieve good performance and increasing S does not greatly improve the performance but introduces significant time/memory overheads.

Algorithm 1 Randomized Attention (RA)

```
Input: the randomized mapping $\xi(\cdot,\cdot)$, queries $\mathbf{Q} := \{\mathbf{q}_n\}_{n=1}^N$, keys $\mathbf{K} := \{\mathbf{k}_m\}_{m=1}^M$, values $\mathbf{V} := \{\mathbf{v}_m\}_{m=1}^M$ and the number of samples $S$;
Output: attention output $\mathbf{Y} := \{\mathbf{y}_n\}_{n=1}^N$;
for $n = 1$ to $N$ do
  for $m = 1$ to $M$ do
    Compute $\pi_{nm} \leftarrow \frac{\exp(\mathbf{q}_n^\top \mathbf{k}_m)}{\sum_{m'=1}^M \exp(\mathbf{q}_n^\top \mathbf{k}_{m'})}$;
  end for
  for $s = 1$ to $S$ do
    Sample $\mathbf{a}_{ns} \sim \text{Categorical}(\boldsymbol{\pi}_n)$;
    Sample $\epsilon_s \sim \mathcal{N}(\epsilon; 0, \mathbf{I})$;
    Compute $\omega_{ns} \leftarrow \mathbf{K}\mathbf{a}_{ns} + \mathbf{q}_n + \epsilon_s$;
    Compute $N_{ns} \leftarrow \xi(\mathbf{q}_n, \omega_{ns}) \sum_{m=1}^M \xi(\mathbf{k}_m, \omega_{ns})\mathbf{v}_m^\top$;
    Compute $D_{ns} \leftarrow \xi(\mathbf{q}_n, \omega_{ns}) \sum_{m=1}^M \xi(\mathbf{k}_m, \omega_{ns})$;
  end for
  Compute $\mathbf{y}_n \leftarrow \frac{1}{S}\sum_{s=1}^S N_{ns}/D_{ns}$;
end for
Return $\mathbf{Y} := [\mathbf{y}_1, \ldots, \mathbf{y}_N]$.
```

A biased variant of randomized attention. Exact sampling from the mixture distribution requires us to first select a discrete component index $\mathbf{a}$ from the mixture distribution and then sample from the corresponding component. Although such randomness might bring additional regularization effect, randomly selecting an index could lead to large variance and slow down training. To accelerate convergence, we also develop a biased sampling strategy from the Gaussian mixture. According to Equation 30, the sampled one-hot vector $\mathbf{a}_n$ can be approximated by its expected value $\boldsymbol{\pi}_n$:

$$\omega_n' = \mathbf{K}\boldsymbol{\pi}_n + \mathbf{q}_n + \epsilon, \quad \epsilon \sim \mathcal{N}(\omega; 0, \mathbf{I}).$$

This introduces a non-negligible sampling bias in estimating the softmax attention; however, it eliminates the need to randomly draw discrete indexing vectors $\mathbf{a}_n$ and reduces the variance, especially in the case of long sequences. In fact, this biased sample can be equivalently viewed as drawn from a Gaussian:

$$\omega' \sim \mathcal{N}(\omega; \mathbf{K}\boldsymbol{\pi}_n + \mathbf{q}_n, \mathbf{I}). \tag{31}$$

Another advantage is that this formulation allows us to maintain fully deterministic during the evaluation mode, while not introducing large discrepancies from training time. Specifically, during evaluation we only pass the expectation $\mathbf{K}\boldsymbol{\pi}_n + \mathbf{q}_n$ as the "sample", which is a standard practice similar to the usage of Dropout (Srivastava et al., 2014). This is in contrast to unbiased RA sampling, which has to draw random indices even during evaluation (otherwise, replacing both random variables with their expected values would lead to larger discrepancies between training and testing, resulting in inferior performance). Same as the case of RFA, we also redraw random samples at every training iteration. Note that this can not be transferred to Performer since the expectation of ω in RFA is 0, which leads to degeneration.

Table 5. Experimental results with exact or biased randomized attention mechanisms.

Model	Complexity	Image		Video		Language
		Top-1 Acc. on ImageNet1k w/ DeiT-Tiny	Top-1 Acc. on ImageNet1k w/ DeiT-Small	Top-1 Acc. on K400	Top-1 Acc. on SSv2	BLEU on WMT
RA-unbiased	Quadratic	71.86	80.04	78.2	64.9	**27.8**
RA-biased	Quadratic	**72.98**	**80.49**	79.0	65.9	27.3
Softmax	Quadratic	72.20	79.90	**79.2**	**66.5**	27.5

As a proof-of-concept experiment, we run randomized attention with biased sampling strategy on image classification with ImageNet1k dataset, video recognition with K400 and SSv2 datasets and machine translation with WMT dataset. From Table 5, we note that biased RA performs better than both its unbiased counterpart for visual tasks, which usually deal with longer sequences (196 for images and 1568 for videos); but it performs worse in machine translation, where either the source or target sentence only consist of dozens of tokens. On the other hand, RA outperforms softmax attention on both image and language tasks, indicating that the proposed estimation methods for softmax attention may enjoy better modeling capacity. This may shed light on some latent mechanism in such approximation that deviates from the standard softmax attention but does better in modeling the sequence representations. We leave detailed investigation in future work.

G.3. Specifics of Linear Randomized Attention

In this section, we provide more implementation details of linear randomized attention.

G.3.1. On the Formulation of Proposal Distributions

As mentioned in §4.1, each proposal $q_c(\omega)$ is defined to depend on some subset of queries; and their union covers the whole set of queries. Since our goal is let these proposals behave similarly to the true RA distribution $p_n(\omega)$, a straightforward choice is to specify q_c as the same formulation of $p_n(\omega)$ (Equation 29):

$$q_c(\omega) = \sum_{m=1}^{M} \frac{\exp(\widetilde{\mathbf{q}}_c^\top \mathbf{k}_m)}{\sum_{m'=1}^{M} \exp(\widetilde{\mathbf{q}}_c^\top \mathbf{k}_{m'})} \mathcal{N}(\omega; \widetilde{\mathbf{q}}_c + \mathbf{k}_m, \mathbf{I}). \tag{32}$$

Here we divide the input query sequence $\{\mathbf{q}_n\}_{n=1}^N$ into C segments and compute the average (called *landmarks*, the number of which is equal to the number of samples) over queries $\{\widetilde{\mathbf{q}}_c\}_{c=1}^C$ within the same segment. In particular, supposing N is divisible by C and $T := N/C$ is the segment length, each segment landmark can be expressed as

$$\widetilde{\mathbf{q}}_c = \frac{1}{T}\sum_{t=1}^{T} \mathbf{q}_{(c-1)T+t}.$$

We then use each of these proposals to estimate the target expectation for the n-th query and combine their results into the final estimation. However, this choice involves CM distributions in total (C proposals are maintained, each of which is again a Gaussian mixture with M components) and sampling from these distributions may introduce large noise. Motivated by the discussion of biased sampling in RA (Equation 31 in Appendix G.2), we explore an alternative parameterization by defining each proposal as a Gaussian:

$$q_c(\omega) = \mathcal{N}(\omega; \mathbf{K}\boldsymbol{\pi}_c + \widetilde{\mathbf{q}}_c, \mathbf{I}) = \mathcal{N}\left(\omega; \widetilde{\mathbf{q}}_c + \sum_{m=1}^{M} \frac{\exp(\widetilde{\mathbf{q}}_c^\top \mathbf{k}_m)}{\sum_{m'=1}^{M} \exp(\widetilde{\mathbf{q}}_c^\top \mathbf{k}_{m'})}\mathbf{k}_m, \mathbf{I}\right). \tag{33}$$

We find this choice performs better than the mixture formulation (Equation 32) empirically. Intuitively, this strategy aggregates the information from all keys based on the correlation between the query landmarks and each individual key. However, this introduces additional $\mathcal{O}(CM)$ computational costs.

In practice, we observe that for proposal landmark $\widetilde{\mathbf{q}}_c$, keys belonging to the same segment c often contribute the most to the Gaussian mean. As a result, we develop another variant that also computes the key landmarks,

$$\widetilde{\mathbf{k}}_c = \frac{1}{T}\sum_{t=1}^{T} \mathbf{k}_{(c-1)T+t},$$

and then simply let

$$q_c(\omega) = \mathcal{N}(\omega; \widetilde{\mathbf{q}}_c + \widetilde{\mathbf{k}}_c, \mathbf{I}). \tag{34}$$

We observe this formulation works equally well; such parameterization is thus used throughout our experiments by default.

An improved proposal parameterization for vision transformers. Comparing Equation 33 and Equation 34, we observe that for the former it only biases the Gaussian mean towards the direction of the current query landmark; while for the latter it only promotes information from key vectors that are in the same segment as $\widetilde{\mathbf{q}}_c$ and ignores the global information of keys. Noticing these differences, we further propose a variant bridging these two formulations:

$$q_c(\omega) = \mathcal{N}\left(\omega; \widetilde{\mathbf{q}}_c + \sum_{c'=1}^{C} \frac{\exp(\widetilde{\mathbf{k}}_c^\top \widetilde{\mathbf{k}}_{c'})}{\sum_{c'=1}^{M} \exp(\widetilde{\mathbf{k}}_c^\top \widetilde{\mathbf{k}}_{c'})} \widetilde{\mathbf{k}}_c, \mathbf{I}\right). \tag{35}$$

Intuitively, this performs an attention-like aggregation operation over key landmarks. The aggregation procedure not only computes the correlation between key vectors, which alleviates the bias of being closer to query landmarks, but also collects global information while still favoring local segments. In addition, it runs with $\mathcal{O}(C^2)$, which is much cheaper than $\mathcal{O}(CM)$. We find this yields better predictive performance in vision transformers, but improves marginally for other tasks. We hypothesize that this is because the attention-like operation smooths the Gaussian mean, which aligns with that ViT tends to produce smoothed patch representations. We leave in-depth investigation as future work. In summary, we adopt this parameterization only through experiments on image classification (§5.2).

See Algorithm 3 for a algorithm sketch of LARA.

Algorithm 2 Random Feature Attention (RFA)

```
Input: the randomized mapping $\xi(\cdot,\cdot)$, queries $\mathbf{Q} := \{\mathbf{q}_n\}_{n=1}^N$, keys $\mathbf{K} := \{\mathbf{k}_m\}_{m=1}^M$, values $\mathbf{V} := \{\mathbf{v}_m\}_{m=1}^M$ and the number of samples $S$;
Output: attention output $\mathbf{Y} := \{\mathbf{y}_n\}_{n=1}^N$;

for $s = 1$ to $S$ do

    Sample $\omega_s \sim \mathcal{N}(\omega; 0, \mathbf{I})$;
    Compute $N_s \leftarrow \sum_{m=1}^M \xi(\mathbf{k}_m, \omega_s)\mathbf{v}_m^\top$;
    Compute $D_s \leftarrow \sum_{m=1}^M \xi(\mathbf{k}_m, \omega_s)$;
end for
for $n = 1$ to $N$ do


    Compute $N \leftarrow \sum_{s=1}^S \xi(\mathbf{q}_n, \omega_s) N_s$;
    Compute $D \leftarrow \sum_{s=1}^S \xi(\mathbf{q}_n, \omega_s) D_s$;
    Compute $\mathbf{y}_n \leftarrow N/D$;
end for
Return $\mathbf{Y} := [\mathbf{y}_1, \dots, \mathbf{y}_N]$.
```

Algorithm 3 Linear Randomized Attention (LARA)

```
Input: the randomized mapping $\xi(\cdot,\cdot)$, queries $\mathbf{Q} := \{\mathbf{q}_n\}_{n=1}^N$, keys $\mathbf{K} := \{\mathbf{k}_m\}_{m=1}^M$, values $\mathbf{V} := \{\mathbf{v}_m\}_{m=1}^M$ and the number of samples $C$;
Output: attention output $\mathbf{Y} := \{\mathbf{y}_n\}_{n=1}^N$;
Compute proposal parameters $\{\mu_c\}_{c=1}^C$;
for $c = 1$ to $C$ do
    Let $q_c(\omega) \leftarrow \mathcal{N}(\omega; \mu_c, \mathbf{I})$;
    Sample $\omega_c \sim q_c(\omega)$;
    Compute $N_c \leftarrow \sum_{m=1}^M \xi(\mathbf{k}_m, \omega_c)\mathbf{v}_m^\top$;
    Compute $D_c \leftarrow \sum_{m=1}^M \xi(\mathbf{k}_m, \omega_c)$;
end for
for $n = 1$ to $N$ do
    Compute $\alpha_{nc}(\omega_c)$ according to Equation 9;
    Compute $\alpha'_{nc}(\omega_c) \leftarrow \alpha_{nc}(\omega_c)\mathcal{N}(\omega_c; 0, \mathbf{I}/q_c(\omega_c))$;
    Compute $N \leftarrow \sum_{c=1}^C \alpha'_{nc}(\omega_c)\xi(\mathbf{q}_n, \omega_c) N_c$;
    Compute $D \leftarrow \sum_{c=1}^C \alpha'_{nc}(\omega_c)\xi(\mathbf{q}_n, \omega_c) D_c$;
    Compute $\mathbf{y}_n \leftarrow N/D$;
end for
Return $\mathbf{Y} := [\mathbf{y}_1, \dots, \mathbf{y}_N]$.
```

G.3.2. On the Parameterization of Weighting Functions

Our MIS estimating strategy introduces a set of weighting functions $\alpha(\cdot)$ for each proposal. A common choice of weighting functions in MIS (Owen, 2013) is the *balance heuristic* strategy

$$\alpha_c(\omega_c) = \frac{q_c(\omega_c)}{\sum_{c'=1}^C q_{c'}(\omega_c)}, \tag{36}$$

which is nearly optimal in that any other weighting schemes will not exhibit significantly smaller variance (Veach & Guibas, 1995). However, this strategy only considers the relative strengths of proposals and ignores contextual information from each query. As a result, a naïve application of MIS would disregard the inherent variation among different queries and fails to describe the specialized target distribution $p_n(\omega)$.

Instead of balance heuristics, we adopt query-specific weighting functions that are inspired by query-optimal analysis. In our MIS scheme (Equation 9), the optimal weighting functions take the following form

$$\alpha^*_{nc}(\omega_c) = \underbrace{\frac{q_c(\omega_c)}{\sum_{c'=1}^C q_{c'}(\omega_c)}}_{\text{balance heuristic}} + \underbrace{q_c(\omega_c)\left(r_{nc}(\omega_c) - \sum_{c=1}^C \frac{q_c(\omega_c)}{\sum_{c'=1}^C q_{c'}(\omega_c)} r_{nc}(\omega_c)\right)}_{\text{query-specific correction}}$$

Note that it sums to 1 over all c's and is a valid weighting function:

$$\begin{aligned}
\sum_{c=1}^C \alpha^*_{nc}(\omega_c) &= \sum_{c=1}^C \frac{q_c(\omega_c)}{\sum_{c'=1}^C q_{c'}(\omega_c)} + \sum_{c=1}^C q_c(\omega_c)\left(r_{nc}(\omega_c) - \sum_{c''=1}^C \frac{q_{c''}(\omega_{c''})}{\sum_{c'=1}^C q_{c'}(\omega_c)} r_{nc''}(\omega_{c''})\right) \\
&= 1 + \left[\sum_{c=1}^C q_c(\omega_c) r_{nc}(\omega_c) - \left(\sum_{c=1}^C q_c(\omega_c)\right) \sum_{c''=1}^C \frac{q_{c''}(\omega_{c''})}{\sum_{c'=1}^C q_{c'}(\omega_c)} r_{nc''}(\omega_{c''})\right] \\
&= 1 + \left[\sum_{c=1}^C q_c(\omega_c) r_{nc}(\omega_c) - \sum_{c=1}^C q_c(\omega_c) r_{nc}(\omega_c)\right] \\
&= 1 + 0 = 1.
\end{aligned}$$

In particular, we observe the first term is the ordinary balance heuristic weighting function, while the second term is a query-specific correction that sums to 0.

As mentioned in Appendix D, the exact form of $r_{nc}(\cdot)$ is mostly intractable to compute in practice. To this end, we introduce a heuristic yet tractable r'_{nc} to roughly align with the intuition of original $r_{nc}(\cdot)$:

$$\alpha_{nc}(\omega_c) = \underbrace{\frac{q_c(\omega_c)}{\sum_{c'=1}^C q_{c'}(\omega_c)}}_{\text{balance heuristic}} + \underbrace{q_c(\omega_c)\left(r'_{nc} - \sum_{c=1}^C \frac{q_c(\omega_c)}{\sum_{c'=1}^C q_{c'}(\omega_c)} r'_{nc}\right)}_{\text{query-specific correction}} \tag{37}$$

$$r'_{nc} = \frac{\exp\left(\mathbf{q}_n^\top \tilde{\mathbf{q}}_c\right)}{\sum_{n=1}^N \exp\left(\mathbf{q}_n^\top \tilde{\mathbf{q}}_{c'}\right)},$$

Intuitively, we implement r'_{nc} as the normalized similarity between the n-th query and the c-th segment-averaged query vector. In addition, we note that the query-specific information r'_{nc} is influenced by the query-agnostic density q_c, which may be incorrectly suppressed or amplified if the drawn sample lies in a low-density region. Base on this, we further propose a simplified formulation:

$$\alpha_{nc}(\omega_c) = \underbrace{\frac{q_c(\omega_c)}{\sum_{c'=1}^C q_{c'}(\omega_c)}}_{\text{balance heuristic}} + \underbrace{r'_{nc} - \frac{1}{C}\sum_{c=1}^C r'_{nc}}_{\text{query-specific correction}}. \tag{38}$$

where we decouple the computation between proposal densities $q_c(\cdot)$ and r'_{nc}. In this way, query-dependent and query-agnostic information will be independent of each other.

We also notice that the query-specific information can be explicitly controlled by introducing a coefficient β such that

$$\alpha_{nc}(\omega_c) = \underbrace{\frac{q_c(\omega_c)}{\sum_{c'=1}^C q_{c'}(\omega_c)}}_{\text{balance heuristic}} + \beta\underbrace{\left(r'_{nc} - \frac{1}{C}\sum_{c=1}^C r'_{nc}\right)}_{\text{correction}}.$$

This weighting function remains valid since the correction term still sums to 0. By setting $\beta > 1$, the mechanism tends to favor the query-specific information over the balance heuristic. We tried several choices of β and found $\beta = 2$ slightly improves the performance. As reflected in our ablation study, we demonstrate the superior performance of query-specific weighting functions over vanilla balance heuristics (Veach & Guibas, 1995).

G.3.3. Training and Evaluation Details

LARA redraws samples from proposal sets at every training iteration; during evaluation, we simply pass corresponding expected values instead of drawing samples, in a similar way to dropout (Srivastava et al., 2014).

G.3.4. Complexity Analysis

Recall there are N queries and M key-value pairs. Like RFA, the involved computation of our LARA estimator includes (1) computing the proposal distribution, which may take $\mathcal{O}(C)$ or $\mathcal{O}(C^2)$ time (Appendix G.3.1); (2) a pre-computing step over all key-value statistics, which takes $\mathcal{O}(CM)$ time and space; and (3) applying pre-computed statistics to all queries, taking $\mathcal{O}(CN)$ complexity. These steps result in overall $\mathcal{O}(CM + CN)$ complexity given $C \ll \min(M, N)$. Note that C is analogous to the number of samples S (often referred to as random feature dimension (Choromanski et al., 2021)) in RFA. Therefore, LARA does not incur a heavy computational overhead compared to RFA, as also reflected in §5.5.

H. Additional Experimental Details

H.1. Preliminary Experiments on Approximation Quality

We conduct the preliminary experiment on vision transformers (ViT), which first split input images into small patches, serialize them as a 1D sequence and then processes the sequence through a transformer model. To be specific, we replace the standard softmax attention in vision transformers (ViT; Dosovitskiy et al., 2021; Touvron et al., 2021) with different approximation variants. The MSE is evaluated under three different sequence lengths N by varying the image resolution and patch size: (a) resolution 224 x 224 with patch size 16 ($N = 196$), (b) resolution 384 x 384 with patch size 16 ($N = 576$) and (c) resolution 224 x 224 with patch size 8 ($N = 784$). To achieve a fair comparison, we use pretrained ViTs whose weights are trained under corresponding sequence lengths with *standard softmax attention*. The sequence length is selected according to whether the ViT weights pretrained by softmax attention are available. Since there are multiple attention blocks in ViT architecture, for each input image we average the attention MSE over all attention heads and transformer layers.

H.2. Image Classification

For image classification, we consider two vision transformer architectures: vanilla ViT (Dosovitskiy et al., 2021) and PVTv2 (Wang et al., 2021b). We refer to ViT as DeiT (Touvron et al., 2021) through this work, since DeiT follows the same model architecture as ViT but adopts greatly improved training protocols.

Details of applying LARA to DeiT. We do not use the distillation technique as in DeiT (Touvron et al., 2021). We following the same procedure to train DeiT on `ImageNet1k` dataset as in Touvron et al. (2021). In particular, we use AdamW optimizer (Loshchilov & Hutter, 2019) for 300 epochs, where we set the batch size to 1024 and the learning rate to 0.001 with cosine learning rate decay (Loshchilov & Hutter, 2016). The number of warm-up epochs is set to 10 for all models instead of 5, since we find it often stabilizes training and leads to better results. For data augmentation, we follow Touvron et al. (2021) and use random clipping, cropping, rand-augment (Cubuk et al., 2020) and random erasing (Zhong et al., 2020). We remove repeated augmentation (Hoffer et al., 2020) as it often slows down convergence, as also observed in previous studies (Berman et al., 2019; Xiao et al., 2021). For regularization, we employ stochastic depth (Huang et al., 2016), Mixup (Zhang et al., 2017), Cutmix (Yun et al., 2019), label smoothing and weight decay, all of which are set to default settings in DeiT (Touvron et al., 2021). Unless otherwise specified, the input image size is set to 224×224 with patch size 16, resulting in $14 \times 14 = 196$ non-overlapping patch tokens. For LARA in DeiT models, we additionally transform the average query/key vector of each segment through a fully connected layer followed by a layer-norm operation. This corresponds to importance sampling with adaptive proposals (Owen, 2013), which improves the expressiveness of the proposal distributions. Note that the linear transformation is shared among all attention heads, which results in only marginal additional computational overheads.

Details of applying LARA to PVTv2. Pyramid Vision Transformers v2 (PVTv2; Wang et al., 2021b) is a strong vision transformer baseline with pyramidal architectures that processes much longer token sequences. It first patchifies input images into a 56×56 token sequence, which is then processed by 4 successive stages. Each stage consists of a stack of transformer layers and processes the input sequence from the previous stage by reducing both the height and width of patch tokens to the half and increasing the embedding dimension by a factor of 2. The detailed configuration for all model sizes follows Table 1 of Wang et al. (2021b). In such architecture, the sequence at early stages is too long to be handled by regular softmax attention. To address this issue, PVTv2 proposes an efficient variant Spatial Reduction Attention (SRA) and uses SRA for all attention blocks in the first three stages and ordinary softmax attention for the last stage due to reduced resolution. For each SRA module, it use a convolutional layer to reduce the length of input sequence to 49, which is then projected to key and value vectors correspondingly. The query set maintains the same resolution and performs attention over the shortened key-value sequence to obtain globally contextualized representations.

To evaluate our method on PVTv2, we replace all SRA modules with either Performer or LARA. For PVTv2 with Performer, we use 128 samples since it fails to converge with fewer samples. In terms of PVTv2 with LARA, we do not use convolutional blocks and simply use 2D average pooling (the same as segments) followed by a linear projection to obtain query and key landmarks, the number of which is set to 49 as in SRA. Since we do not use the convolutional block, Both Performer and LARA use much fewer model parameters than vanilla PVTv2.

In addition, vanilla PVTv2 model uses 1,2,5 and 8 attention heads for its 4 processing stages respectively in its original implementation; however, we found using $2\times$ more heads consistently improves predictive performance for all methods (including baseline SRA, Performer and LARA) while introducing affordable overheads. Therefore, we use 2,4,10 and 16 heads for all PVTv2-based models across our experiments. We mostly follow the training protocol as Wang et al. (2021b) to train all PVTv2-based models, except that we increase the number of warm-up epochs from 5 to 10. We find a slightly longer warm-up schedule is helpful to improve the model performance.

H.3. Video Action Recognition

Our implementation is based on the `PySlowFast` (Fan et al., 2020) codebase and we follow the training protocol in Motionformer (Patrick et al., 2021). In particular, Motionformer adopts the vision transformer base (ViT/B) (Dosovitskiy et al., 2021) architecture which has 12 transformer encoder layers with 12 attention heads and 768-dimensional hidden representations. For `K400` dataset, its parameter weights are pretrained on `ImageNet21k` dataset with regular softmax attention; while for `SSv2` dataset, we use the trained weights on `K400` with the corresponding attention variant. The model operates on videos with size $16 \times 224 \times 224$, which is then split into $8 \times 14 \times 14$ tubes with separate space-time positional embedding. Motionformer introduces the trajectory attention, which first computes spatial attention to obtain probabilistic trajectories, which are then aggregated temporally. We use the trajectory attention module (Patrick et al., 2021) and replace the involved softmax attention with different attention approximation methods. Besides Performer (Choromanski et al., 2021), Nyströmformer (Xiong et al., 2021) and full trajectory attention, our baselines also include Orthoformer (Patrick et al., 2021), another strong baseline for video transformers that constructs a low-rank approximation of attention matrix via sequentially selecting orthogonal query landmarks. For all efficient attention variants, we set both the number of samples (in LARA and Performer) or the number of landmarks (in Nyströmformer and Orthoformer) to 128 for a fair comparison. For data augmentation, we also follow Patrick et al. (2021), adopting random scale jittering, random horizontal flips and color jittering for all datasets; and additionally rand-augment (Cubuk et al., 2020) for `SSv2` dataset.

We use the AdamW (Loshchilov & Hutter, 2019) optimizer to train LARA for 40 epochs with weight decay 0.05, label smoothing rate 0.2 and total batch size 256. A slightly longer training schedule (compared to 35) is adopted since our method involves additional randomness and we found training for a longer time improves convergence. The initial learning rate is set to 0.0001 and gets decayed by a ratio of 10 at epochs 25 and 35 respectively. During training, the video clips are randomly sampled with cropped resolution 224×224; while for testing, we sample 10 uniform temporal clips per video with 3 spatial crops per clip and average scores for these crops to obtain the final prediction.

H.4. Machine Translation

We use the Transformer-base architecture as specified in Vaswani et al. (2017) for our machine translation experiments. The model contains a transformer encoder and decoder, both of which consist of 6 layers with hidden size and number of heads being 512 and 8, respectively. The vocabulary is shared between source and target language, consisting of around 32K byte pair encoding (BPE; Sennrich et al., 2016) types. The hidden dimension of feed forward networks is set to 2048. The rate of dropout is set to 0.1. As mentioned in the main paper, we only replace encoder self-attention in transformer models with efficient attention variants. Since LARA does not support causal attention mode in its current version, this setting allows us to directly assess the ability of different attention mechanisms to learn contextualized representations. Recent studies also indicate that in neural machine translation the transformer encoder seems playing a more important role in extracting representations (Kasai et al., 2021a). Besides Performer, we also compare our method against other baselines including (1) Linformer (Wang et al., 2020), which is widely adopted in the context of NLP, (2) ABC (Peng et al., 2021a), a recently

Table 6. Empirical running time and memory consumption for different attention mechanisms. We report the absolute time in millisecond and memory usage in GB; the relative time/memory cost to the softmax attention is also reported in brackets.

Models	Running Time (ms)					Peak Memory Usage (GB)				
	1024	2048	3072	4096	8192	1024	2048	3072	4096	8192
Full softmax	9.44 (1.00×)	19.96 (1.00×)	42.45 (1.00×)	69.25 (1.00×)	271.12 (1.00×)	0.11 (1.00×)	0.33 (1.00×)	0.68 (1.00×)	1.18 (1.00×)	4.58 (1.00×)
Nyströmformer	37.51 (3.98×)	36.90 (1.85×)	37.44 (0.88×)	37.21 (0.54×)	38.36 (0.14×)	0.05 (0.45×)	0.06 (0.20×)	0.08 (0.12×)	0.10 (0.08×)	0.16 (0.03×)
Linformer	12.85 (1.36×)	12.57 (0.63×)	12.57 (0.30×)	12.62 (0.18×)	19.14 (0.07×)	0.05 (0.46×)	0.07 (0.20×)	0.08 (0.12×)	0.10 (0.09×)	0.17 (0.04×)
Performer	18.74 (1.99×)	18.80 (0.94×)	19.04 (0.45×)	18.89 (0.27×)	33.00 (0.12×)	0.05 (0.44×)	0.06 (0.19×)	0.08 (0.11×)	0.09 (0.08×)	0.15 (0.03×)
LARA	19.91 (2.11×)	19.82 (0.99×)	19.87 (0.47×)	19.91 (0.29×)	31.81 (0.12×)	0.06 (0.51×)	0.08 (0.24×)	0.10 (0.15×)	0.12 (0.10×)	0.19 (0.04×)
RA	17.09 (1.81×)	53.56 (2.68×)	108.17 (2.55×)	183.03 (2.64×)	756.72 (2.79×)	0.23 (2.10×)	0.80 (2.45×)	1.74 (2.55×)	3.06 (2.59×)	12.10 (2.64×)

proposed unified framework of most low-rank attention approximations and (3) Nyströmformer (Xiong et al., 2021), which we find is a strong low-rank baseline across our experiments.

For training, we follow the same setup as in Vaswani et al. (2017). In particular, we use the Adam optimizer (Kingma & Ba, 2014) with learning rate 0.0007, label smoothing rate 0.1, inverse square root learning rate scheduler and 4000 warm-up steps. During decoding, we set beam size to 4, length penalty to 0.6, average last 10 checkpoints and apply a compound split post-processing to facilitate comparison.

H.5. Efficiency Analysis

For the simulation experiment conducted in §5.5, the same transformer architecture is used for all attention methods, which consists of 8 encoder layers with 192 embedding dimension and 3 attention heads. The use of smaller-size transformer model allows us to run longer lengths for softmax attention and randomized attention. The detailed running time (in ms) and memory consumption is listed in Table 6. For Nyströmformer (Xiong et al., 2021) and Linformer (Wang et al., 2020), the number of landmarks is set to 16; for Performer and LARA, the number of samples is set to 16 as well.

I. Additional Experimental Results

I.1. Additional Experiments on Image Classification

We conduct additional experiments to evaluate the performance of our proposed method at various aspects. First, we vary the number of random samples to investigate the effect of sample size on the performance for `ImageNet1k` dataset. As presented in Table 7, although both Performer and LARA improves predictive accuracy as the number of samples increases, LARA benefits much more than Performer, and finally outperforms softmax attention with 196 samples, which is equal to the sequence length.

In addition, we also compare LARA against different efficient attention mechanisms, as shown in Table 8. We note that LARA outperforms most efficient attention mechanisms by a large margin, and slightly outperforms Nyströmformer (Xiong et al., 2021), which we found is a strong baseline across various domains.

Table 7. Top-1 accuracy (%) on `ImageNet1k` validation set for Performer and Lara at different numbers of samples on DeiT-Small.

Model	# Samples			
	25	49	100	196
Performer	73.37	73.63	74.15	74.44
LARA	78.29	79.48	79.89	80.57
RA	80.04			
Softmax	79.90			

Table 8. Top-1 accuracy (%) on `ImageNet1k` validation set with DeiT-Small under different attention mechanisms.

Model	Top-1 Acc.
Performer (Choromanski et al., 2021)	74.3
SRA (Convolutional) (Wang et al., 2021a;b)	74.4
Linformer (Wang et al., 2020)	76.0
XCIT (El-Nouby et al., 2021)	77.9
Nyströmformer (Xiong et al., 2021)	79.3
LARA	**79.5**
Softmax attention	79.9

I.2. Additional Experiments on Long Range Arena Benchmark

We also evaluate our model on the Long Range Arena (`LRA`) benchmark (Tay et al., 2021b), which is designed to test the ability to process long sequences and generalize over diverse tasks. In particular, `LRA` is a suite of tasks including

Table 9. Top-1 classification accuracy results (%) on LRA benchmark with different attention mechanisms.

Model	ListOps	Text	Retrieval	Image	Pathfinder	Avg.
Softmax	38.76	64.90	80.54	39.90	71.29	59.08
Nyströmformer	38.26	64.00	80.57	40.07	68.47	58.27
Linformer	37.40	59.10	78.04	38.25	60.09	54.58
Reformer	37.60	64.15	79.18	43.57	66.44	58.19
BigBird	38.81	64.02	80.73	38.56	71.60	58.74
Performer	37.20	64.73	79.91	37.64	68.68	57.63
LARA	39.21	64.77	81.18	38.40	72.02	59.12
RA	38.56	65.02	80.93	40.76	71.22	**59.30**

Table 10. Ablation study of LARA, evaluated on ImageNet1k validation set with DeiT-Tiny model.

Components	Variants	Top-1 Acc.
Performer		65.92
Single or multiple proposals	Single proposal	68.42
	Multiple proposals	71.48
Weighting functions	Balance heuristic	70.78
	Approx. optimal (coupled; Equation 37)	71.02
	Approx. optimal (decoupled; Equation 38)	71.48
Parameterization of each proposal	Gaussian mixture (Equation 32)	70.22
	Gaussian (Equation 33)	71.22
	Gaussian (Equation 34)	71.19
	Gaussian (Equation 35)	71.48

Listops output prediction (Nangia & Bowman, 2018), byte-level text classification on IMDb (Maas et al., 2011), byte-level document retrieval on AAN (Radev et al., 2013), pixel-level image recognition on CIFAR-10 (Krizhevsky et al., 2009) and Pathfinder (Linsley et al., 2018). We follow the experimental setup in Xiong et al. (2021); Chen et al. (2021d) and adopt the *same* hyper-parameter setting across all attention variants to ensure a fair comparison. In particular, all tasks use a 2-layer Transformer model with 64 embedding dimension, 128 hidden dimension in feed forward neural networks and 2 attention heads. The transformer output is then aggregated by mean pooling (instead of class tokens) for task-specific prediction. The training details for each task are the same for all attention methods as in Xiong et al. (2021). For baselines, we compare our model against the standard softmax attention and Performer (Choromanski et al., 2021) as well as other efficient attention mechanisms, including Nyströmformer (Xiong et al., 2021), Linformer (Wang et al., 2020), Reformer (Kitaev et al., 2020) and BigBird (Zaheer et al., 2020).

As shown in Table 9, we see that RA performs better than softmax attention on 3 out of 5 tasks and obtains a higher averaged accuracy. Furthermore, its linear-complexity counterpart LARA also performs competitively with or slightly outperforms softmax attention except the image task. Both RA and LARA yield better performance than Performer and other baselines on all of 5 tasks, indicating the improved expressiveness of our proposed method. As the sequence length considered in this suite of tasks is typically longer, these results also validates the ability of RA and LARA to capture longer-term dependencies.

I.3. Ablation Study

In this section, we conduct an ablation study on vision transformers with ImageNet1k dataset to investigate the effect of component design in LARA. The main component design choices in LARA consist of the estimation framework (single proposal versus multiple proposals), the parameterization of proposal distributions (Gaussian mixtures versus Gaussian) and the weighting functions. The results are shown in Table 10. For the estimation framework, we compare our choice, which uses multiple proposal distributions, against a single proposal. This proposal is a Gaussian mixture with the similar formulation of true RA density (Equation 5) except that it only depends on the average of all queries. We see that an individual yet contextual proposal improves the performance of Performer, while generalizing the importance sampling in

RFA to multiple proposals further boosts performance to be close to softmax attention. With multiple proposal distributions, even using a simple strategy (balance heuristic (Veach & Guibas, 1995)) to combine their estimates yields reasonable performance, which is improved further by adopting query-specific combinations. In addition, we validate the effectiveness of decoupling the effect of query-dependent and query-agnostic information inside the weighting function, which improves the coupled version by over 0.4 accuracy. In terms of the parameterization of each proposal distribution, we consider both the cases where each proposal is a Gaussian mixture and a Gaussian. As specified in Appendix G.3, we train the transformer model with various parameterization choices (defined by Equation 32 for Gaussian mixtures and Equations 33 to 35 for Gaussians). The results are consistent with the analysis in Appendix G.3, where a simple parameterization suffices to yield good performance.