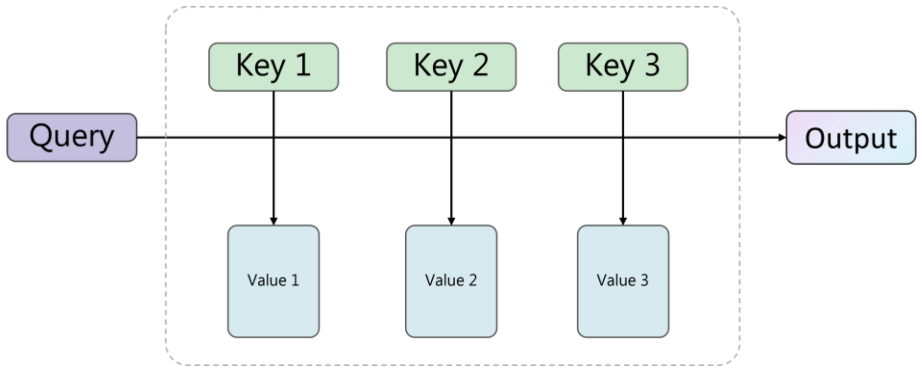
Linear Complexity Randomized Self-attention Mechanism

Lin Zheng, Chong Wang, Lingpeng Kong

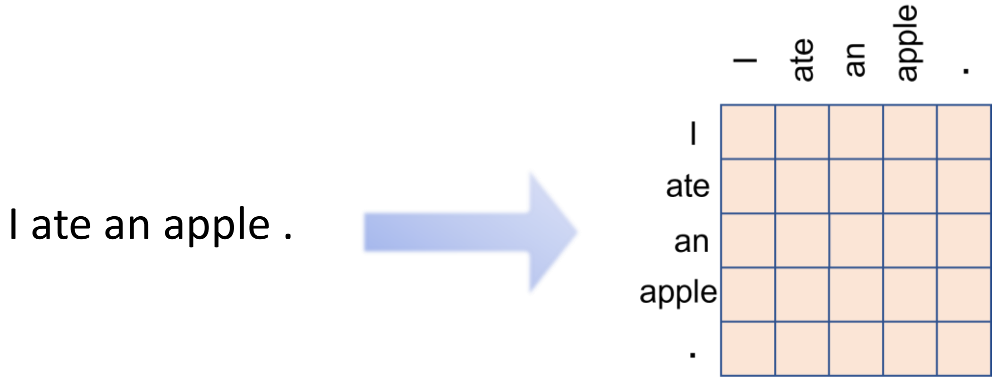
Attention



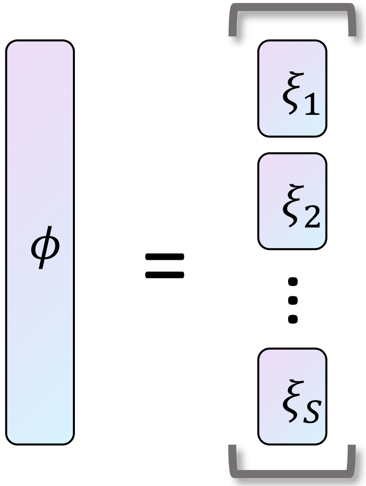
$$\mathrm{Attn}\left(\mathbf{q}_n, \{\mathbf{k}_m\}, \{\mathbf{v}_m\}\right) = \sum_m \frac{\exp\left(\mathbf{q}_n^\top \mathbf{k}_m\right)}{\sum_{m'} \exp(\mathbf{q}_n^\top \mathbf{k}_{m'})} \mathbf{v}_m^\top$$

✔ Effective in capturing long-range dependencies and yielding contextualized representations.

✘ Running with **quadratic** complexity; prohibitive to process long sequences.



Random Feature-based Attention



- The key idea is to decompose the exponential kernel into a dot-product of **random features**:

$$\exp(\mathbf{x}^\top \mathbf{y}) = \mathbb{E}_{\omega \sim \mathcal{N}(0,\mathbf{I})} \left[\xi(\mathbf{x}, \omega)^\top \xi(\mathbf{y}, \omega) \right] \approx \frac{1}{S} \sum_{s=1}^{S} \xi(\mathbf{x}, \omega^s)^\top \xi(\mathbf{y}, \omega^s) := \phi\left(\mathbf{q}_n, \boldsymbol{\omega}\right)^\top \phi\left(\mathbf{k}_m, \boldsymbol{\omega}\right)$$

- Throughout this work we consider positive random features:

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

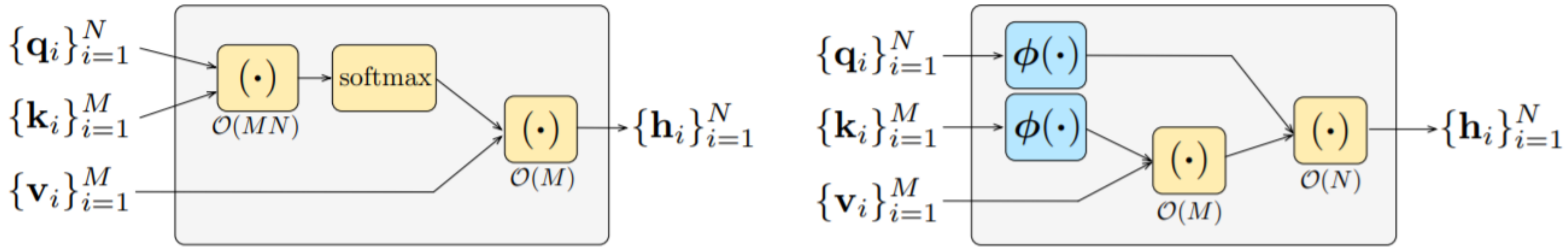
- Plugging in such approximation yields RFA:

$$\sum_m \frac{\exp\left(\mathbf{q}_n^\top \mathbf{k}_m\right)}{\sum_{m'} \exp\left(\mathbf{q}_n^\top \mathbf{k}_{m'}\right)} \mathbf{v}_m^\top \approx \sum_m \frac{\phi\left(\mathbf{q}_n, \boldsymbol{\omega}\right)^\top \phi\left(\mathbf{k}_m, \boldsymbol{\omega}\right) \mathbf{v}_m^\top}{\sum_{m'} \phi\left(\mathbf{q}_n, \boldsymbol{\omega}\right)^\top \phi\left(\mathbf{k}_{m'}, \boldsymbol{\omega}\right)} := \mathrm{RFA}\left(\mathbf{q}_n, \{\mathbf{k}_m\}, \{\mathbf{v}_m\}\right)$$

Random feature attention

- RFA achieves **linear** complexity due to the re-order of computation.
- Reduce complexity from $\mathcal{O}(MN)$ to $\mathcal{O}(M + N)$.

$$\sum_m \frac{\phi(\mathbf{q}_n, \boldsymbol{\omega})^\top \phi(\mathbf{k}_m, \boldsymbol{\omega}) \mathbf{v}_m^\top}{\sum_{m'} \phi(\mathbf{q}_n, \boldsymbol{\omega})^\top \phi(\mathbf{k}_{m'}, \boldsymbol{\omega})} = \frac{\phi(\mathbf{q}_n, \boldsymbol{\omega})^\top \sum_{m=1}^{M} \phi(\mathbf{k}_m, \boldsymbol{\omega}) \otimes \mathbf{v}_m}{\phi(\mathbf{q}_n, \boldsymbol{\omega})^\top \sum_{m'=1}^{M} \phi(\mathbf{k}_{m'}, \boldsymbol{\omega})}$$



(a) Softmax attention. (b) Random feature attention.

Figure from RFA paper: https://arxiv.org/abs/2103.02143

What goes wrong?

- Despite its efficiency, RFA suffers from **poor** modeling performance and slow training convergence.
- To investigate this, we observe that although the approximation to each exponential kernel is unbiased, the approximation to the whole attention is biased!
- This is due to the non-linearity of ratios.

$$\sum_m \frac{\exp\left(\mathbf{q}_n^\top \mathbf{k}_m\right)}{\sum_{m'} \exp\left(\mathbf{q}_n^\top \mathbf{k}_{m'}\right)} \mathbf{v}_m^\top = \sum_m \frac{\mathbb{E}\left[\phi\left(\mathbf{q}_n, \boldsymbol{\omega}\right)^\top \phi\left(\mathbf{k}_m, \boldsymbol{\omega}\right)\right] \mathbf{v}_m^\top}{\sum_{m'} \mathbb{E}\left[\phi\left(\mathbf{q}_n, \boldsymbol{\omega}\right)^\top \phi\left(\mathbf{k}_{m'}, \boldsymbol{\omega}\right)\right]} \not\approx \sum_m \frac{\phi\left(\mathbf{q}_n, \boldsymbol{\omega}\right)^\top \phi\left(\mathbf{k}_m, \boldsymbol{\omega}\right) \mathbf{v}_m^\top}{\sum_{m'} \phi\left(\mathbf{q}_n, \boldsymbol{\omega}\right)^\top \phi\left(\mathbf{k}_{m'}, \boldsymbol{\omega}\right)}$$

This work

- *Question: we already know how to unbiasedly estimate exponential kernels. But how do we estimate the whole softmax attention in an unbiased manner?*

$$\exp(\mathbf{q}_n^\top \mathbf{k}_m)\mathbf{v}_m^\top = \mathbb{E}_\omega\left[\xi(\mathbf{q}_n,\omega)^\top \xi(\mathbf{k}_m,\omega)\mathbf{v}_m^\top\right] \qquad \text{(previous work)}$$

$$\sum_m \frac{\exp\left(\mathbf{q}_n^\top \mathbf{k}_m\right)}{\sum_{m'} \exp\left(\mathbf{q}_n^\top \mathbf{k}_{m'}\right)}\mathbf{v}_m^\top = \mathbb{E}_\omega\left[?\right] \qquad \text{(our work)}$$

An Alternative View of Softmax Attention

- We prove that softmax attention can be written as an expectation over RFA-like functions:

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

  - $f_n(\omega)$ is an RFA-like aggregating function:

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

  - $p_n(\omega)$ is a Gaussian mixture with input-dependent parameters:

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

Randomized Attention (RA)

- This results readily implies an **unbiased** estimator to the whole softmax attention:

$$
\begin{aligned}
\mathrm{SoftmaxAttn}(\mathbf{q}_n, \{\mathbf{k}_m\}, \{\mathbf{v}_m\}) &= \mathbb{E}_{p_n(\omega)}\left[\frac{\sum_{m=1}^{M} \xi(\mathbf{q}_n,\omega)^\top \xi(\mathbf{k}_m,\omega)\mathbf{v}_m}{\sum_{m'=1}^{M} \xi(\mathbf{q}_n,\omega)^\top \xi(\mathbf{k}_{m'},\omega)}\right] \\
&\approx \frac{1}{S}\sum_{s=1}^{S} \frac{\sum_{m=1}^{M} \xi(\mathbf{q}_n,\omega_n^s)^\top \xi(\mathbf{k}_m,\omega_n^s)\mathbf{v}_m}{\sum_{m'=1}^{M} \xi(\mathbf{q}_n,\omega_n^s)^\top \xi(\mathbf{k}_{m'},\omega_n^s)} \\
&:= \mathrm{RandAttn}(\mathbf{q}_n, \{\mathbf{k}_m\}, \{\mathbf{v}_m\})
\end{aligned}
$$

- Here $\omega_n^1, \ldots, \omega_n^S \sim p_n(\omega)$. We call the resulting estimator **Randomized Attention** (RA).
- To the best of our knowledge, this is the first unbiased estimator to softmax attention in terms of kernel linearization.

RFA as an SNIS estimator

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

- Furthermore, we show that RFA is equivalent to a **self-normalized importance sampler** to approximate softmax attention,

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

- with the proposal distribution $\omega^s \sim q(\omega) = \mathcal{N}(0, \mathbf{I})$.

Comparing RA and RFA

- We have two estimators available: RA (unbiased) and RFA (biased).
- RA is **more effective** than RFA:
  - ✔ is **adaptive** and **query-specific**;
  - ✔ processes sequences at a finer-grained level.

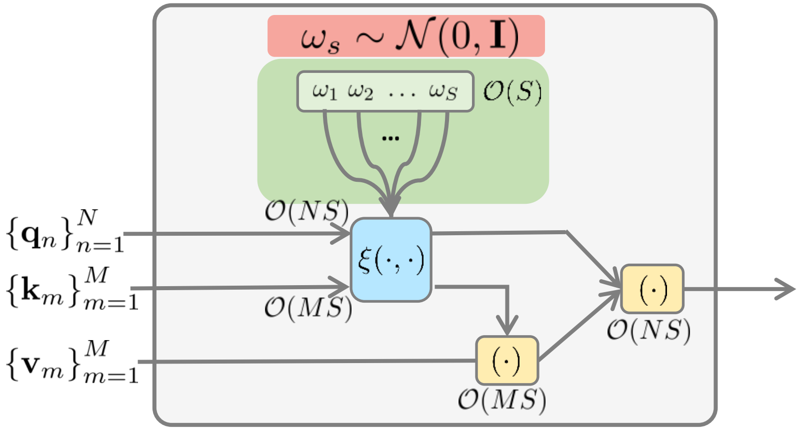
$$\text{RA:} \quad \omega_n \sim p(\omega) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\omega; \mathbf{q}_n + \mathbf{k}_m, \mathbf{I})$$

$$\text{RFA:} \quad \omega \sim q(\omega) = \mathcal{N}(\omega; 0, \mathbf{I})$$

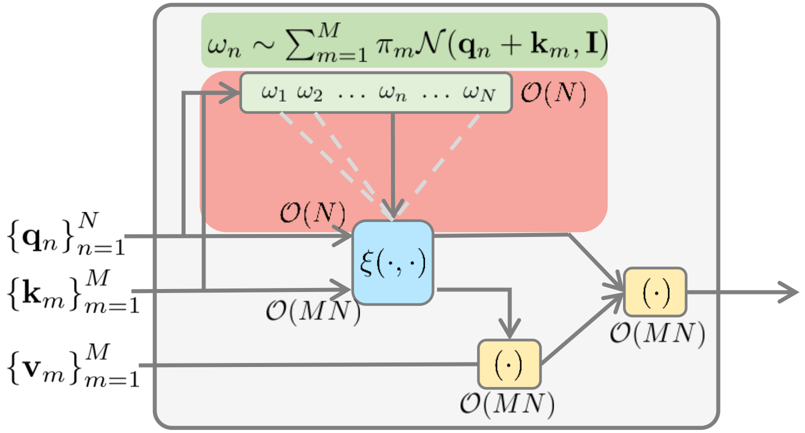
- RA is **less efficient** than RFA:
  - ✘ Samples ω_n from a query-dependent distribution, making $\xi(k,\omega_n)$ distinct for different queries.
  - ✘ As a result, even though we can reorder computation, it still requires quadratic complexity!

$$\sum_m \frac{\xi(\mathbf{q}_n, \boldsymbol{\omega}_n)^\top \xi(\mathbf{k}_m, \boldsymbol{\omega}_n)\, \mathbf{v}_m^\top}{\sum_{m'} \xi(\mathbf{q}_n, \boldsymbol{\omega}_n)^\top \xi(\mathbf{k}_{m'}, \boldsymbol{\omega}_n)} = \frac{\xi(\mathbf{q}_n, \boldsymbol{\omega}_n)^\top \sum_{m=1}^{M} \xi(\mathbf{k}_m, \boldsymbol{\omega}_n) \otimes \mathbf{v}_m}{\xi(\mathbf{q}_n, \boldsymbol{\omega}_n)^\top \sum_{m'=1}^{M} \xi(\mathbf{k}_{m'}, \boldsymbol{\omega}_n)}$$

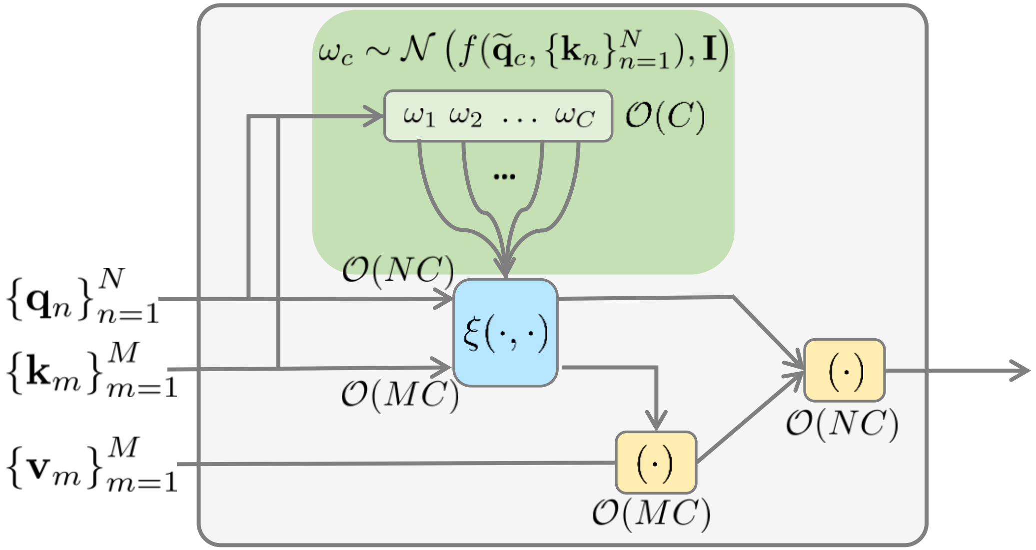
LARA: Linear randomized attention



RFA



RA



LARA

LARA: Linear randomized attention

- We propose LARA, a linear complexity attention that combines both the expressiveness of RA and the efficiency of RFA.
- To remain efficiency
  - **Self-normalized importance sampling** formulation is kept to **share** proposal distributions among queries.
- To improve expressiveness
  - **Adaptive** multiple proposal distributions (beyond simple standard Gaussians as in RFA) are used and combined in a **query-specific** way.

LARA: Linear randomized attention

- The resulting approximation to softmax attention, called **Linear randomized attention** or LARA, has a concise formulation:

$$\mathrm{LARA}\left(\mathbf{q}_n, \{\mathbf{k}_m\}, \{\mathbf{v}_m\}\right) = \frac{\sum_{c=1}^{C} \alpha'_{nc}(\omega_c) \sum_{m=1}^{M} \xi(\mathbf{q}_n, \omega_c)^\top \xi(\mathbf{k}_m, \omega_c)\mathbf{v}_m}{\sum_{c=1}^{C} \alpha'_{nc}(\omega_c) \sum_{m'=1}^{M} \xi(\mathbf{q}_n, \omega_c)^\top \xi(\mathbf{k}_{m'}, \omega_c)}, \quad \omega_c \sim q_c(\omega)$$

- Comparing with RFA:

$$\mathrm{RFA}\left(\mathbf{q}_n, \{\mathbf{k}_m\}, \{\mathbf{v}_m\}\right) = \frac{\sum_{s=1}^{S} \sum_{m=1}^{M} \xi(\mathbf{q}_n, \omega^s)^\top \xi(\mathbf{k}_m, \omega^s)\mathbf{v}_m}{\sum_{s=1}^{S} \sum_{m'=1}^{M} \xi(\mathbf{q}_n, \omega^s)^\top \xi(\mathbf{k}_{m'}, \omega^s)}, \quad \omega^s \sim \mathcal{N}(0, \mathbf{I})$$

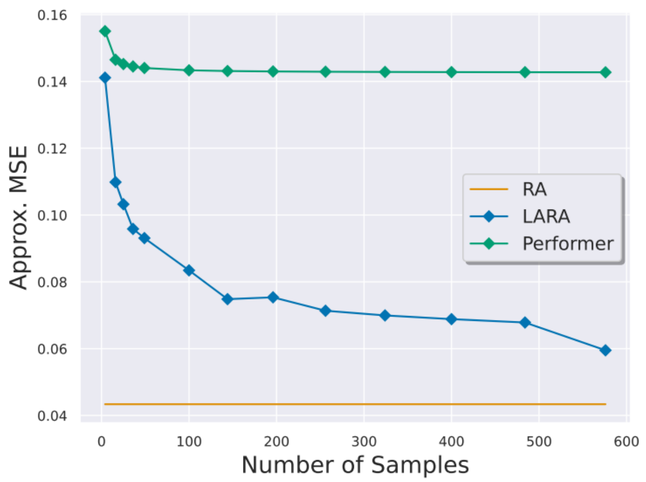
Experiments

- LARA improves vanilla RFA (such as Performer) by a large margin, and performs competitively with the unbiased RA.
- It scales better to longer sequences or more samples.
- It works well even with only a few of samples (e.g., <32), unlike vanilla RFA (which typically requires $\mathcal{O}(d)$ samples; d is the vector size).

Results of applying LARA to ViTs

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

Approximation error to softmax attention



Experiments

- LARA outperforms most previous efficient attention mechanisms.
- When applied to advanced ViT architectures, LARA achieves SOTA results.

Image classification results

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

Machine translation results

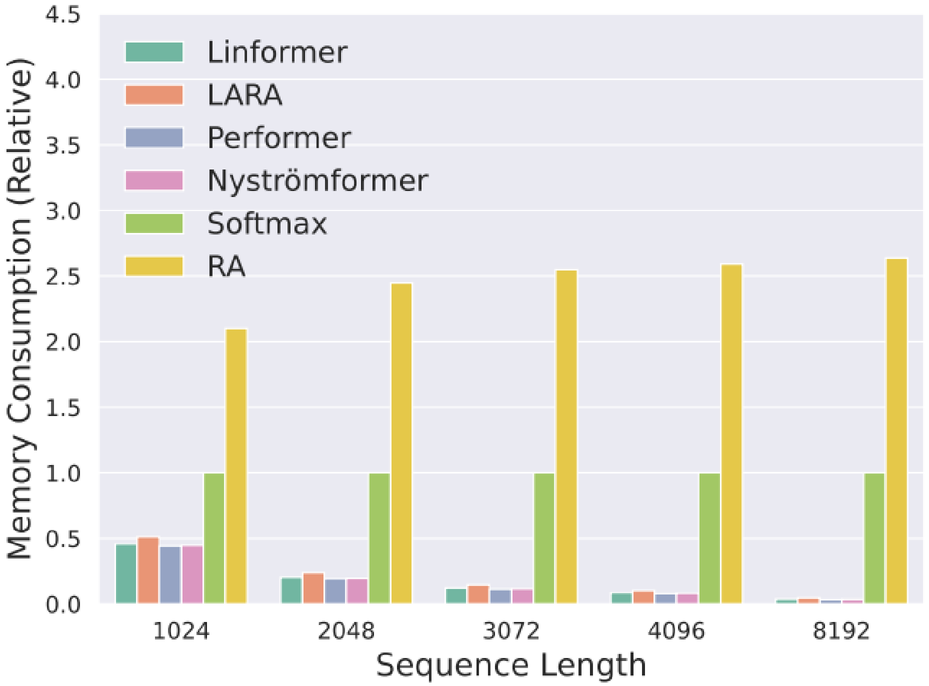
Model	# samples	# Param.	BLEU
Softmax	n.a.	60.92M	27.5
ABC	16	60.93M	25.4
	32	60.94M	25.6
	64	60.95M	26.0
Linformer	16	60.92M	17.4
	32	61.31M	23.0
	64	61.70M	23.7
Nyströmformer	16	60.92M	25.1
	32	60.92M	26.8
	64	60.92M	26.8
Performer	64	60.92M	–
	128	60.92M	23.5
	256	60.92M	23.7
	512	60.92M	23.3
LARA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

Classification results on ImageNet1k dataset compared with SOTA models.

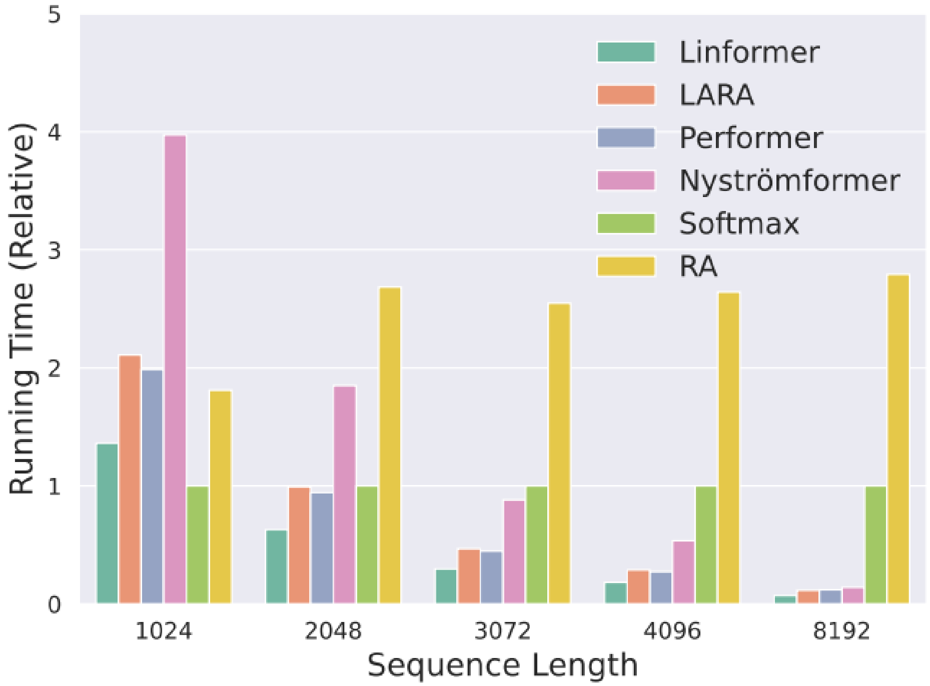
Model	# Param.	FLOPs	Top-1 Acc.
PVT-v1-T (Wang et al., 2021a)	13.2M	2.1G	75.1
SOFT-T (Lu et al., 2021)	13.1M	1.9G	79.3
RegionViT-T (Chen et al., 2021b)	13.8M	2.4G	**80.4**
PVT-v2-b1 (SRA)	14.0M	2.1G	78.7
PVT-v2-b1 + Performer	12.1M	2.5G	77.3
PVT-v2-b1 + LARA	13.7M	2.3G	79.6
PVT-v1-S (Wang et al., 2021a)	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**

Experiments

- LARA incurs little additional memory consumption and running time compared to vanilla RFA (Performer).



Memory consumption



Running time

Thanks!