RANDOM FEATURE ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers are state-of-the-art models for a variety of sequence modeling tasks. At their core is an attention function which models pairwise interactions between the inputs at every timestep. While attention is powerful, it does *not* scale efficiently to long sequences due to its quadratic time and space complexity in the sequence length. We propose RFA, a linear time and space **a**ttention that uses **r**andom **f**eature methods to approximate the softmax function, and explore its applications in transformers. RFA offers a straightforward way of learning with recency bias through an optional gating mechanism and can be used as a drop-in replacement for conventional softmax attention. Experiments on language modeling and machine translation demonstrate that RFA achieves similar or better performance compared to strong transformer baselines. In the machine translation experiment, RFA decodes twice as fast as a vanilla transformer. Our analysis shows that RFA's efficiency gains are especially notable on long sequences, suggesting that RFA will be particularly useful in tasks that require working with large inputs, fast decoding speed, or low memory footprints.

1 INTRODUCTION

Transformer architectures (Vaswani et al., 2017) have achieved tremendous success on a variety of sequence modeling tasks (Ott et al., 2018; Radford et al., 2018; Parmar et al., 2018; Devlin et al., 2019; Parisotto et al., 2019, *inter alia*). Under the hood, the key component is attention (Bahdanau et al., 2015). Attention models pairwise interactions of the inputs, regardless of their distance. This comes with quadratic time and memory costs, making the transformers computationally expensive, especially for long sequences. A large body of research has been devoted to improving their time and memory efficiency (Tay et al., 2020b). Some are able to achieve better *asymptotic* complexity (Lee et al., 2019; Child et al., 2019; Sukhbaatar et al., 2019; Beltagy et al., 2020, *inter alia*), while it is more challenging to improve on shorter sequences: the additional computation steps required by some approaches can overshadow the time and memory they save (Kitaev et al., 2020; Wang et al., 2020; Roy et al., 2020, *inter alia*).

This work proposes **r**andom **f**eature **a**ttention (RFA). It scales linearly in sequence length in terms of both time and space. RFA builds on a kernel perspective of softmax (Rawat et al., 2019). Using the well-established random feature maps (Rahimi & Recht, 2008; Avron et al., 2016; §2), RFA approximates the dot-then-exponentiate function with a kernel trick (Hofmann et al., 2008): $\exp(\mathbf{x} \cdot \mathbf{y}) \approx \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{y})$. Inspired by its connections to gated recurrent neural networks (Hochreiter & Schmidhuber, 1997; Cho et al., 2014) and differentiable plasticity (Ba et al., 2016; Miconi et al., 2018), we further augment RFA with an optional gating mechanism, offering a straightforward way of learning with recency bias when locality is desired.

RFA and its gated variant (§3) can be used as a drop-in substitute for the canonical softmax attention, and increase the number of parameters by less than 0.1%. We explore its applications in transformers on language modeling and machine translation (§4). Our experiments show that RFA achieves comparable performance to vanilla transformer baselines in both tasks, while outperforming a recent related approach (Katharopoulos et al., 2020). The gating mechanism proves particularly useful in language modeling: the gated variant of RFA outperforms the transformer baseline on WikiText-103. RFA shines in decoding, even for shorter sequences. In our head-to-head comparison on machine translation benchmarks, RFA decodes around 2× faster than a transformer baseline, *without* accuracy loss. Our analysis shows that more significant time and memory efficiency improvements can be achieved for longer sequences: 12× decoding speedup with less than 10% of the memory for 2,048-length outputs.

2 Background

2.1 Attention in Sequence Modeling

The attention mechanism (Bahdanau et al., 2015) has been widely used in many sequence modeling tasks. Its dot-product variant is the key building block for the state-of-the-art transformer architectures (Vaswani et al., 2017). Let $\{\mathbf{q}_t\}_{t=1}^N$ denote a sequence of N **query** vectors, that attend to sequences of M **key** and **value** vectors. At each timestep, the attention linearly combines the values weighted by the outputs of a softmax:

$$\operatorname{attn}\left(\mathbf{q}_t, \{\mathbf{k}_i\}, \{\mathbf{v}_i\}\right) = \sum_i \frac{\exp\left(\mathbf{q}_t \cdot \mathbf{k}_i / \tau\right)}{\sum_j \exp\left(\mathbf{q}_t \cdot \mathbf{k}_j / \tau\right)} \mathbf{v}_i^\top. \tag{1}$$

τ is the temperature hyperparameter determining how "flat" the softmax is (Hinton et al., 2015).[1]

Calculating attention for a single query takes $\mathcal{O}(M)$ time and space. For the full sequence of N queries the space amounts to $\mathcal{O}(MN)$. When the computation *cannot* be parallelized across the queries, e.g., in autoregressive decoding, the time complexity is quadratic in the sequence length.

2.2 Random Feature Methods

The theoretical backbone of this work is the unbiased estimation of the Gaussian kernel by Rahimi & Recht (2008). Based on Bochner's theorem (Bochner, 1955), Rahimi & Recht (2008) proposed random Fourier features to approximate a desired shift-invariant kernel. The method nonlinearly transforms a pair of vectors $\mathbf{x}$ and $\mathbf{y}$ using a **random feature map** $\boldsymbol{\phi}$; the inner product between $\boldsymbol{\phi}(\mathbf{x})$ and $\boldsymbol{\phi}(\mathbf{y})$ approximates the kernel evaluation on $\mathbf{x}$ and $\mathbf{y}$. More precisely:

Theorem 1 (Rahimi & Recht, 2008)**.** *Let* $\boldsymbol{\phi} : \mathbb{R}^d \rightarrow \mathbb{R}^{2D}$ *be a nonlinear transformation:*

$$\boldsymbol{\phi}\left(\mathbf{x}\right) = \sqrt{1/D}\Big[\sin\left(\mathbf{w}_1 \cdot \mathbf{x}\right), \ldots, \sin\left(\mathbf{w}_D \cdot \mathbf{x}\right), \cos\left(\mathbf{w}_1 \cdot \mathbf{x}\right), \ldots, \cos\left(\mathbf{w}_D \cdot \mathbf{x}\right)\Big]^\top. \tag{2}$$

When d-dimensional random vectors $\mathbf{w}_i$ are independently sampled from $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_d)$,

$$\mathbb{E}_{\mathbf{w}_i}\left[\boldsymbol{\phi}\left(\mathbf{x}\right) \cdot \boldsymbol{\phi}\left(\mathbf{y}\right)\right] = \exp\left(- \left\|\mathbf{x} - \mathbf{y}\right\|^2 / 2\sigma^2\right). \tag{3}$$

Variance of the estimation is inversely proportional to D (Yu et al., 2016).[2]

Random feature methods proved successful in speeding up kernel methods (Oliva et al., 2015; Avron et al., 2017; Sun, 2019, *inter alia*), and more recently are used to efficiently approximate softmax (Rawat et al., 2019). In §3.1, we use it to derive an unbiased estimate to $\exp(\langle \cdot\, , \cdot \rangle)$ and then an efficient approximation to softmax attention.

3 Model

This section presents RFA (§3.1) and its gated variant (§3.2). In §3.3 we lay out several design choices and relate RFA to prior works. We close by practically analyzing RFA's complexity (§3.4).

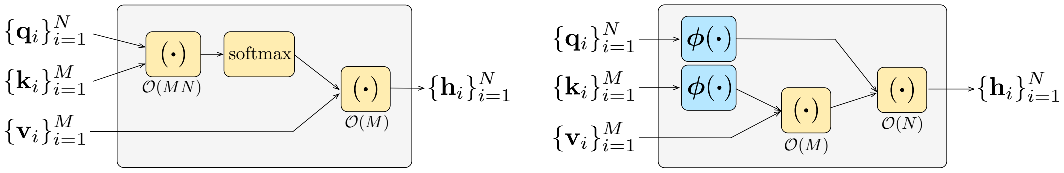
3.1 Random Feature Attention

RFA builds on an unbiased estimate to $\exp(\langle \cdot\, , \cdot \rangle)$ from Theorem 1, which we begin with:

$$\begin{aligned} \exp\left(\mathbf{x} \cdot \mathbf{y} / \sigma^2\right) &= \exp\left(\left\|\mathbf{x}\right\|^2 / 2\sigma^2 + \left\|\mathbf{y}\right\|^2 / 2\sigma^2\right) \exp\left(- \left\|\mathbf{x} - \mathbf{y}\right\|^2 / 2\sigma^2\right) \\ &\approx \exp\left(\left\|\mathbf{x}\right\|^2 / 2\sigma^2 + \left\|\mathbf{y}\right\|^2 / 2\sigma^2\right) \, \boldsymbol{\phi}\left(\mathbf{x}\right) \cdot \boldsymbol{\phi}\left(\mathbf{y}\right). \end{aligned} \tag{4}$$

[1] $M = N$ in self attention; they may differ, e.g., in the cross attention of a sequence-to-sequence model.

[2] In addition, the variance decreases as $\left\|\mathbf{x} - \mathbf{y}\right\| / \sigma$ does (Yu et al., 2016).



$\{\mathbf{q}_i\}_{i=1}^N$
$\{\mathbf{k}_i\}_{i=1}^M$
$\{\mathbf{v}_i\}_{i=1}^M$
$(\cdot)$
$\mathcal{O}(MN)$
softmax
$(\cdot)$
$\mathcal{O}(M)$
$\{\mathbf{h}_i\}_{i=1}^N$
$\{\mathbf{q}_i\}_{i=1}^N$
$\{\mathbf{k}_i\}_{i=1}^M$
$\{\mathbf{v}_i\}_{i=1}^M$
$\boldsymbol{\phi}(\cdot)$
$\boldsymbol{\phi}(\cdot)$
$(\cdot)$
$\mathcal{O}(M)$
$(\cdot)$
$\mathcal{O}(N)$
$\{\mathbf{h}_i\}_{i=1}^N$

(a) Softmax attention.

(b) Random feature attention.

Figure 1: Computation graphs for softmax attention (left) and random feature attention (right). Here, we assume cross attention with source length M and target length N.

The last line does *not* have any nonlinear interaction between $\boldsymbol{\phi}(\mathbf{x})$ and $\boldsymbol{\phi}(\mathbf{y})$, allowing for a linear time/space approximation to softmax attention. For clarity we assume the query and keys are unit vectors.[3] $\operatorname{attn}(\mathbf{q}_t, \{\mathbf{k}_i\}, \{\mathbf{v}_i\}) =$

$$\begin{aligned}\sum_i \frac{\exp\left(\mathbf{q}_t \cdot \mathbf{k}_i / \sigma^2\right)}{\sum_j \exp\left(\mathbf{q}_t \cdot \mathbf{k}_j / \sigma^2\right)} \mathbf{v}_i^\top &\approx \sum_i \frac{\boldsymbol{\phi}\left(\mathbf{q}_t\right)^\top \boldsymbol{\phi}\left(\mathbf{k}_i\right) \mathbf{v}_i^\top}{\sum_j \boldsymbol{\phi}\left(\mathbf{q}_t\right) \cdot \boldsymbol{\phi}\left(\mathbf{k}_j\right)} \\ &= \frac{\boldsymbol{\phi}\left(\mathbf{q}_t\right)^\top \sum_i \boldsymbol{\phi}\left(\mathbf{k}_i\right) \otimes \mathbf{v}_i}{\boldsymbol{\phi}\left(\mathbf{q}_t\right) \cdot \sum_j \boldsymbol{\phi}\left(\mathbf{k}_j\right)} = \textsc{Rfa}\left(\mathbf{q}_t, \{\mathbf{k}_i\}, \{\mathbf{v}_i\}\right).\end{aligned} \tag{5}$$

$\otimes$ denotes the outer product between vectors, and σ^2 corresponds to the temperature term τ in Eq. 1.

RFA can be used as a drop-in-replacement for softmax-attention. The latter is typically used in two different ways in the transformer architecture, each resulting in a different computation for RFA:

(a) **Causal attention** attends to the prefix.[4] Its computation is similar to that in RNNs: tuple $(\mathbf{S}_t \in \mathbb{R}^{D \times d}, \mathbf{z}_t \in \mathbb{R}^d)$ can be seen as the hidden state at time step t:

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{\phi}\left(\mathbf{k}_t\right) \otimes \mathbf{v}_t, \quad \mathbf{z}_t = \mathbf{z}_{t-1} + \boldsymbol{\phi}\left(\mathbf{k}_t\right), \quad \mathbf{h}_t^\top = \boldsymbol{\phi}\left(\mathbf{q}_t\right)^\top \mathbf{S}_t / \boldsymbol{\phi}\left(\mathbf{q}_t\right) \cdot \mathbf{z}_t. \tag{6}$$

(b) The input is revealed in full to **cross attention** and **encoder self attention**. The context is first "squashed" into a matrix-vector tuple $(\mathbf{S}, \mathbf{z})$, and then queried by $\mathbf{q}_t$:

$$\mathbf{S} = \sum_t \boldsymbol{\phi}\left(\mathbf{k}_t\right) \otimes \mathbf{v}_t, \quad \mathbf{z} = \sum_t \boldsymbol{\phi}\left(\mathbf{k}_t\right), \quad \mathbf{h}_t^\top = \boldsymbol{\phi}\left(\mathbf{q}_t\right)^\top \mathbf{S} / \boldsymbol{\phi}\left(\mathbf{q}_t\right) \cdot \mathbf{z}. \tag{7}$$

Outputs $\{\mathbf{h}_t\}$ are used for onward computation. Algos. 1 and 2 (Appendix A) summarize the computation procedure of RFA, and Figures 1 compare it against the softmax attention.

Analogously to the softmax attention, RFA has its multiheaded variant (Vaswani et al., 2017). Note that in the sequence-to-sequence case, the input to RFA cross-attention is of fixed-length size as opposed to variable-length size in the standard transformer. In the experiments we use causal RFA in a transformer language model (§4.1), and both cross and causal RFA in the decoder of a sequence-to-sequence machine translation model.

3.2 RFA-GATE: LEARNING WITH RECENCY BIAS

The canonical softmax attention does *not* have any explicit modeling of distance or locality. In learning problems where such inductive bias is crucial (Ba et al., 2016; Parmar et al., 2018; Miconi et al., 2018; Li et al., 2019, *inter alia*), transformers heavily rely on positional encodings. Answering to this, many approaches have been proposed, e.g., learning the attention spans (Sukhbaatar et al., 2019; Wu et al., 2020), and enhancing the attention computation with recurrent (Hao et al., 2019; Chen et al., 2019) or convolutional (Wu et al., 2019; Mohamed et al., 2019) components.

RFA faces the same issue, but its causal attention variant (Eq. 6) offers a straightforward way of learning with recency bias. We draw inspiration from its connections to RNNs, and augment RFA

[3] This can be achieved by ℓ_2-normalizing the query and keys. See §3.3 for a related discussion.

[4] It is also sometimes called "decoder self-attention" or "autoregressive attention."

with a learned gating mechanism (Hochreiter & Schmidhuber, 1997; Cho et al., 2014):

$$\begin{aligned} g_t &= \text{sigmoid}(\mathbf{w}_g \cdot \mathbf{x}_t + b_g), \\ \mathbf{S}_t &= g_t \, \mathbf{S}_{t-1} + (1 - g_t) \, \boldsymbol{\phi}(\mathbf{k}_t) \otimes \mathbf{v}_t, \\ \mathbf{z}_t &= g_t \, \mathbf{z}_{t-1} + (1 - g_t) \, \boldsymbol{\phi}(\mathbf{k}_t). \end{aligned} \tag{8}$$

$\mathbf{w}_g$ and b_g are learned parameters, and $\mathbf{x}_t$ is the input representation at timestep t.[5] By multiplying the learned scalar gates $0 < g_t < 1$ against the hidden state $(\mathbf{S}_t, \mathbf{z}_t)$, history is exponentially decayed, favoring more recent context.

The gating mechanism shows another benefit of RFA: it would be otherwise more difficult to build similar techniques into the softmax attention, where there is no clear sense of "recurrence." It proves useful in our language modeling experiments (§4.1).

3.3 Discussion

On query and key norms, and learned random feature variance. Eq. 5 assumes both the query and keys are of norm-1. It therefore approximates a softmax attention that normalizes the queries and keys before multiplying them, and then scales the logits by dividing them by σ^2. Empirically, this normalization step scales down the logits (Vaswani et al., 2017) and enforces that $-1 \leq \mathbf{q}^\top \mathbf{k} \leq 1$. A consequence of this is that the softmax outputs would be "flattened" if not for σ, which can be set *a priori* as a hyperparameter (Yu et al., 2016; Avron et al., 2017; Sun, 2019, *inter alia*). Here we instead learn it from data with the reparameterization trick (Kingma & Welling, 2014):

$$\widetilde{\mathbf{w}}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d), \quad \mathbf{w}_i = \boldsymbol{\sigma} \circ \widetilde{\mathbf{w}}_i. \tag{9}$$

$\mathbf{I}_d$ is the $d \times d$ identity matrix, and $\circ$ denotes elementwise product between vectors. d-dimensional vector $\boldsymbol{\sigma}$ is learned.[6]

This norm-1 constraint is never mandatory. Rather, we employ it for notation clarity and easier implementation. In preliminary experiments we find it has little impact on the performance when σ is set properly or learned from data. Eq. 10 in Appendix A presents RFA *without* imposing it.

Going beyond the Gaussian kernel. More broadly, random feature methods can be applied to a family of shift-invariant kernels, with the Gaussian kernel being one of them. In the same family, the order-1 arc-cosine kernel (Cho & Saul, 2009) can be approximated with feature map: $\boldsymbol{\phi}_{\text{arccos}}(\mathbf{x}) = \sqrt{1/D}[\text{ReLU}(\mathbf{w}_1 \cdot \mathbf{x}), \ldots, \text{ReLU}(\mathbf{w}_D \cdot \mathbf{x})]^\top$ (Alber et al., 2017).[7] In our experiments, the Gaussian and arc-cosine variants achieve similar performance. This supplements the exploration of alternatives to softmax in attention (Tsai et al., 2019; Gao et al., 2019).

Relations to prior works. Katharopoulos et al. (2020) inspires the causal attention variant of RFA. They use a feature map based on the the exponential linear units activation (Clevert et al., 2016): $\text{elu}(\cdot) + 1$. It significantly *underperforms* both the baseline and RFA in our controlled experiments, showing the importance of a properly-chosen feature map. Random feature approximation of attention is also explored by a concurrent work (Choromanski et al., 2020), with applications in masked language modeling for proteins. They point out that with further assumptions (that this work does not enforce), such approximation uniformly converges to the softmax attention. Going beyond both, RFA establishes the benefits of random feature methods as a more universal substitute for softmax across all attention variants, facilitating its applications in, e.g., sequence-to-sequence learning.

There are interesting connections between gated RFA and fast weights (Ba et al., 2016) and differentiable plasticity (Miconi et al., 2018). Emphasizing recent patterns, they learn a temporal memory to store history similarly to Eqs. 8. The main difference is that RFA additionally normalizes the output using $\boldsymbol{\phi}(\mathbf{q}_t) \cdot \mathbf{z}$ as in Eq. 6, an output from approximating softmax's partition function. It is intriguing to study the role of this normalization term, which we leave to future work.

[5] In multihead attention (Vaswani et al., 2017), $\mathbf{k}_t$ and $\mathbf{v}_t$ are calculated from $\mathbf{x}_t$ using learned affine transformations.

[6] This departs from Eq. 2 by lifting the isotropic assumption imposed on the Gaussian distribution: note the difference between the vector $\boldsymbol{\sigma}$ in Eq. 9 and the scalar σ in Eq. 2. We find this improves the performance in practice (§4), even though the same result in Theorem 1 may not directly apply.

[7] Apart from replacing the sinusoid functions with ReLU, it constructs $\mathbf{w}_i$ in the same way as Eq. 9.

3.4 Complexity Analysis

Time. Scaling linearly in the sequence lengths, RFA implies speedup wherever the complexity is quadratic for softmax attention. More specifically:

- Significant speedup can be expected in autoregressive *decoding*, both conditional (e.g., machine translation) and unconditional (e.g., sampling from a language model). A $1.9\times$ decoding speedup is achieved with the relatively short outputs in our machine translation experiments (§4.2); and more for longer sequences (e.g., $12\times$ for 2,048-length ones; §5).
- For some applications, the softmax attention can be parallelized across the time steps, and thus hardly any speed improvement can be achieved. These include language modeling, text classification, and teacher forcing training of sequence-to-sequence models.[8]

Memory. Asymptotically, RFA achieves a better memory efficiency than its softmax counterpart (linear vs. quadratic). To reach a more practical conclusion, we include in our analysis the cost of the feature maps. $\boldsymbol{\phi}$'s memory overhead largely depends on its size D. For example, let's consider the cross attention of a decoder. RFA uses $\mathcal{O}(2D + Dd)$ space to store $\boldsymbol{\phi}(\mathbf{q}_t)$, $\mathbf{z}$, and $\mathbf{S}$ (Eqs. 7; line 12 of Algo. 2). In contrast, softmax cross attention stores the encoder outputs with $\mathcal{O}(Md)$ memory, with M being the source length. In this case RFA has a lower memory overhead when $D \ll M$. Typically D should be no less than d in order for reasonable approximation (Yu et al., 2016); In a transformer model, d is the size of an attention head, which is usually around 64 or 128 (Vaswani et al., 2017; Ott et al., 2018). This suggests that RFA can achieve significant memory saving with longer sequences, which is supported by our empirical analysis in §5. Further, using moderate sized feature maps is also desirable, so that its overhead does not overshadow the time and memory RFA saves. We experiment with D at d and $2d$; the benefit of using $D > 2d$ is marginal.

Table 3 in Appendix A discusses the time and space complexity in more detail.

4 Experiments

We evaluate RFA on language modeling and machine translation.

4.1 Language Modeling

We experiment with WikiText-103 (Merity et al., 2016). It is based on English Wikipedia. Table 4 in Appendix B summarizes some of its statistics.

Setting. We compare the following models:

- BASE is our implementation of the strong transformer-based language model by Baevski & Auli (2019).
- RFA builds on BASE, but replaces the softmax attention with random feature attention. We experiment with both Gaussian and arc-cosine kernel variants.
- RFA-GATE additionally learns a sigmoid gate on top of RFA (§3.2). It also has a Gaussian kernel variant and a arc-cosine kernel one.[9]
- $\boldsymbol{\phi}_{\text{elu}}$ is a baseline to RFA. Instead of the random feature methods it uses the $\text{elu}(\cdot) + 1$ feature map, as in Katharopoulos et al. (2020).

To ensure fair comparisons, we use comparable implementations, tuning, and training procedure. All models use a 512 block size during both training and evaluation, i.e., they read as input a segment of 512 consecutive tokens, *without* access to the context from previous mini-batches. RFA variants use 64-dimensional random feature maps. We experiment with two model size settings, **small** (around 38M parameters) and **big** (around 242M parameters); they are described in Appendix B.1 along with other implementation details.

Results. Table 1 compares the models' performance in perplexity on WikiText-103 development and test data. Both kernel variants of RFA, *without* gating, outperform $\boldsymbol{\phi}_{\text{elu}}$ by more than 2.4 and 2.1 test perplexity for the small and big model respectively, confirming the benefits from using random

[8] A common and efficient teacher-forcing training implementation grants the transformer decoder full access to the gold target, but masks future tokens (Ott et al., 2019).

[9] This gating technique is specific to RFA variants, in the sense that it is less intuitive to apply it in BASE.

	Small		**Big**	
Model	**Dev.**	**Test**	**Dev.**	**Test**
BASE	33.0	34.5	24.5	26.2
ϕ_{elu} (Katharopoulos et al., 2020)	38.4	40.1	28.7	30.2
RFA-Gaussian	33.6	35.7	25.8	27.5
RFA-arccos	36.0	37.7	26.4	28.1
RFA-GATE-Gaussian	**31.3**	**32.7**	**23.2**	**25.0**
RFA-GATE-arccos	**32.8**	**34.0**	24.8	26.3
RFA-GATE-Gaussian-Stateful	**29.4**	**30.5**	**22.0**	**23.5**

Table 1: Language model perplexity (**lower is better**) on the WikiText-103 development and test sets. Bolded numbers outperform BASE.

feature approximation.[10] Yet both *underperform* BASE, with RFA-Gaussian having a smaller gap. Comparing RFA against its gated variants, a more than 1.8 perplexity improvement can be attributed to the gating mechanism; and the gap is larger for small models. Notably, RFA-GATE-Gaussian outperforms BASE under both size settings by at least 1.2 perplexity. In general, RFA models with Gaussian feature maps outperform their arc-cosine counterparts.[11] From the analysis in §3.4 we would *not* expect speedup by RFA models, nor do we see any in the experiments.[12]

Closing this section, we explore a "stateful" variant of RFA-GATE-Gaussian. It passes the last hidden state $(\mathbf{S}_t, \mathbf{z}_t)$ to the next mini-batch during both training and evaluation, a technique commonly used in RNN language models (Merity et al., 2018). This is a consequence of RFA's RNN-style computation, and is less straightforward to be applicable in the vanilla transformer models.[13] From the last row of Table 1 we see that this brings a more than 1.5 test perplexity improvement.

4.2 MACHINE TRANSLATION

Datasets. We experiment with three standard machine translation datasets.

- WMT14 EN-DE and EN-FR (Bojar et al., 2014). Our data split and preprocessing follow those of Vaswani et al. (2017). We share the source and target vocabularies within each language pair, with 32,768 byte pair encoding types (BPE; Sennrich et al., 2016).
- IWSLT14 DE-EN (Cettolo et al., 2014) is based on TED talks. The preprocessing follows Edunov et al. (2018). Separate vocabularies of 9K/7K BPE types are used for the source and target.

Table 4 in Appendix B summarizes some statistics of the datasets.

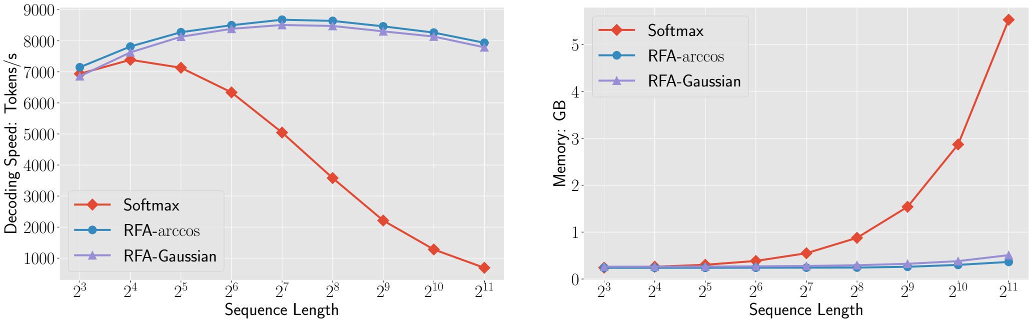
Setting. We compare the RFA variants described in §4.1. Here the BASE model they build on is our implementation of the base-sized transformer by Vaswani et al. (2017). All RFA models apply random feature attention in decoder cross and causal attention, but use softmax attention in encoders. By §3.4, this setting yields the greatest decoding time and memory savings. The cross attention feature maps are 128-dimensional, and the causal ones are 64-dimensional. The gated variants learn sigmoid gates in the decoder causal attention. The ϕ_{elu} baseline uses the same setting and applies feature map in both decoder cross and causal attention, but *not* in the encoders. We

[10] All models are trained for 150K steps; this could be part of the reason behind the suboptimal performance of ϕ_{elu}: it may need 3 times more gradient updates to reach similar performance to the softmax attention baseline (Katharopoulos et al., 2020).

[11] We observe that RFA Gaussian variants are more stable and easier to train than the arc-cosine ones as well as ϕ_{elu}. We conjecture that this is because the outputs of the Gaussian feature maps have an ℓ_2-norm of 1, which can help stabilize training. To see why, $\sin^2(x) + \cos^2(x) = \cos(x - x) = 1$.

[12] In fact, RFA *trains* around 15% slower than BASE due to the additional overhead from the feature maps.

[13] Some transformer models use a text segment from the previous mini-batch as a prefix (Baevski & Auli, 2019; Dai et al., 2019). Unlike RFA, this gives the model access to only a limited amount of context, and significantly increases the memory overhead.



(a) Speed vs. lengths.

(b) Memory vs. lengths.

Figure 2: Conditional decoding speed (left) and memory overhead (right) varying the output lengths. All models are tested on a single TPU v2 accelerator, with greedy decoding and batch size 16.

evaluate the models using SacreBLEU (Post, 2018).[14] A beam search with beam size 4 and length penalty 0.6 is used. Further details are described in Appendix B.2.

	WMT14		**IWSLT14**	
Model	**EN-DE**	**EN-FR**	**DE-EN**	**Speed**
BASE	28.1	39.0	34.6	1.0×
ϕ_{elu} (Katharopoulos et al., 2020)	21.3	34.0	29.9	2.0×
RFA-Gaussian	28.0	39.2	34.5	1.8×
RFA-arccos	28.1	38.9	34.4	1.9×
RFA-GATE-Gaussian	28.1	39.0	34.6	1.8×
RFA-GATE-arccos	28.2	39.2	34.4	1.9×

Table 2: Machine translation test set BLEU. The decoding speed (last column) is relative to BASE. All models are tested on a single TPU v2 accelerator, with batch size 32.

Results. Table 2 compares the models' test set BLEU on three machine translation datasets. Overall both Gaussian and arc-cosine variants of RFA achieve similar performance to BASE on all three datasets, significantly outperforming Katharopoulos et al. (2020). Differently from the trends in the language modeling experiments, here the gating mechanism does not lead to substantial gains. Notably, all RFA variants decode more than 1.8× faster than BASE.

5 ANALYSIS

Decoding time and memory varying sequence lengths. §3.4 shows that RFA can potentially achieve more significant speedup and memory saving for longer sequences, which we now explore.

We use a simulation experiment to compare RFA's sequence-to-sequence decoding speed and memory overhead against the baseline's. Here we assume the input and output sequences are of the same length. The compared models are of the same size as those described in §4.2. All models are tested using greedy decoding with the same batch size of 16.

From Figures 2 (a) and (b) we observe clear trends. Varying the lengths, both RFA variants achieve consistent decoding speed with nearly-constant memory overhead. In contrast, the baseline decodes slower for longer sequences taking an increasing amount of memory. Notably, for 2,048-length sequences, RFA decodes around 12× faster than the baseline while using less than 10% of the memory. These results suggest that RFA can be particularly useful in sequence-to-sequence tasks with longer sequences, e.g., document-level machine translation (Miculicich et al., 2018).

[14] https://github.com/mjpost/sacrebleu

Figure 3 in Appendix C.1 compares the speed and memory consumption in *unconditional* decoding (e.g., sampling from a language model). The overall trends are similar to those in Figure 2.

Notes on decoding speed. With a lower memory overhead, RFA can use a larger batch size than the baseline. As noted by Katharopoulos et al. (2020) and Kasai et al. (2020), if we had used minibatches as large as the hardware allows, RFA would have achieved a more significant speed gain. Nonetheless, we control for batch size even though it is not the most favorable setting for RFA, since the conclusion translates better to common applications where one generates a single sequence at a time (e.g., instantaneous machine translation). For the softmax attention baseline, we follow Ott et al. (2018) and cache previously computed query/key/value representations, which significantly improves its decoding speed (over not caching).

Further analysis results. RFA achieves comparable performance to softmax attention. Appendix C.2 empirically shows that this *cannot* be attributed to that RFA learns a good approximation to softmax: we train with one attention but evaluate with the other, the performance is hardly better than randomly-initialized untrained models. Yet, a RFA model initialized from a pretrained softmax transformer achieves decent training loss after a moderate amount of finetuning steps (Appendix C.3). This suggests some potential applications, e.g., to transfer knowledge from a softmax transformer pretrained on large-scale data (e.g., GPT-3; Brown et al., 2020) to a RFA model that is efficient to sample from.

6 Related Work

One common motivation across the following studies, that is shared by this work and the research we have already discussed, is to scale transformers to very long text, audio, or image sequences. Note that there are plenty orthogonal choices for improving efficiency such as weight sharing (Dehghani et al., 2019), quantization (Shen et al., 2020), knowledge distillation (Sanh et al., 2020), and adapters (Houlsby et al., 2019). For a detailed overview we refer the reader to Tay et al. (2020b).

Sparse attention patterns. The idea behind is to limit the reception filed of attention computation to reduce memory overhead. It motivates one of the earliest attempts in improving attention's efficiency, and still receives lots of interest. The sparse patterns can be set *a priori* (Liu et al., 2018; Qiu et al., 2019; Child et al., 2019; Beltagy et al., 2020; Ho et al., 2020; You et al., 2020, *inter alia*) or learned from data (Sukhbaatar et al., 2019; Roy et al., 2020; Tay et al., 2020a; Kitaev et al., 2020, *inter alia*). For most of these approaches, it is yet to be empirically verified that they are suitable for large-scale sequence-to-sequence learning; few of them have recorded decoding speed benefits.

Compressed context. Wang et al. (2020) propose to compress the context along the timesteps so that the effective sequence length for attention computation is reduced. Another line of work aims to store past context into a memory module with limited size (Lee et al., 2019; Ainslie et al., 2020; Rae et al., 2020, *inter alia*), so that accessing longer history only moderately increase the overhead. Related to these approaches, Dai et al. (2019) use a segment-level recurrence mechanism to attend to a fixed size of past memories, a technique that proves successful in transformer language models. Reminiscent of RNN language models, RFA attends beyond a fixed context window through a stateful computation, *without* increasing time or memory overhead.

To the best of our knowledge, RFA is the first to perform on par with standard transformer on language modeling and machine translation while providing significant decoding time and memory efficiency improvements in practical settings.

7 Conclusion

We presented random feature attention (RFA). It views the softmax attention through the lens of kernel methods, and approximates it with random feature methods. With an optional gating mechanism, RFA provides a straightforward way of learning with recency bias. RFA's time and space complexity is linear in the sequence length. We use RFA as a drop-in substitute for softmax attention in transformer models. On machine translation and language modeling benchmarks, RFA achieves comparable or better performance than strong baselines. In the machine translation experiment, RFA decodes twice faster. Further time and memory efficiency improvements can be achieved for longer sequences.

REFERENCES

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. Etc: Encoding long and structured inputs in transformers, 2020.

Maximilian Alber, Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Fei Sha. An empirical study on the properties of random bases for kernel methods. In *Proc. of NeurIPS*, 2017.

Haim Avron, Vikas Sindhwani, Jiyan Yang, and Michael W. Mahoney. Quasi-monte carlo feature maps for shift-invariant kernels. *Journal of Machine Learning Research*, 17(120):1–38, 2016.

Haim Avron, L. Kenneth Clarkson, and P. David and Woodruff. Faster kernel ridge regression using sketching and preconditioning. *SIAM J. Matrix Analysis Applications*, 2017.

Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *Proc. of NeurIPS*, 2016.

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In *Proc. of ICLR*, 2019.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun (eds.), *Proc. of ICLR*, 2015.

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

S. Bochner. *Harmonic Analysis and the Theory of Probability*. University of California Press, 1955.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proc. of WMT*, 2014.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th IWSLT evaluation campaign. In *Proc. of IWSLT*, 2014.

Kehai Chen, Rui Wang, Masao Utiyama, and Eiichiro Sumita. Recurrent positional embedding for neural machine translation. In *Proc. of EMNLP*, 2019.

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

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. of EMNLP*, 2014.

Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *Proc. of NeurIPS*, 2009.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, David Belanger, Lucy Colwell, and Adrian Weller. Masked language modeling for proteins via linearly scalable long-context transformers, 2020.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs), 2016.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proc. of ACL*, 2019.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *Proc. of ICLR*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*, 2019.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. Classical structured prediction losses for sequence to sequence learning. In *Proc. of NAACL*, 2018.

Yingbo Gao, Christian Herold, Weiyue Wang, and Hermann Ney. Exploring kernel functions in the softmax layer for contextual word classification. In *International Workshop on Spoken Language Translation*, 2019.

Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. Modeling recurrence for transformer. In *Proc. of NAACL*, 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36(3):1171–1220, 2008.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proc. of ICML*, 2019.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A. Smith. Deep encoder, shallow decoder: Reevaluating the speed-quality tradeoff in machine translation, 2020.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Francois Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proc. of ICML*, 2020.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of ICLR*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proc. of ICLR*, 2014.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *Proc. of ICLR*, 2020.

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

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Proc. of NeurIPS*, 2019.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. In *Proc. of ICLR*, 2018.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and Optimizing LSTM Language Models. In *Proc. of ICLR*, 2018.

Thomas Miconi, Kenneth Stanley, and Jeff Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *Proc. of ICML*, 2018.

Lesly Miculicich, Dhananjay Ram, Nikolaos Pappas, and James Henderson. Document-level neural machine translation with hierarchical attention networks. In *Proc. of EMNLP*, 2018.

Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer. Transformers with convolutional context for asr. *arXiv: Computation and Language*, 2019.

Junier Oliva, William Neiswanger, Barnabas Poczos, Eric Xing, Hy Trac, Shirley Ho, and Jeff Schneider. Fast function to function regression. In *Proc. of AISTATS*, 2015.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *Proc. of WMT*, 2018.

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

Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant M. Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, Matthew M. Botvinick, Nicolas Heess, and Raia Hadsell. Stabilizing transformers for reinforcement learning, 2019.

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

Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, 2018.

Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding, 2019.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2018.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *Proc. of ICLR*, 2020.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proc. of NeurIPS*, 2008.

Ankit Singh Rawat, Jiecao Chen, Felix Xinnan X Yu, Ananda Theertha Suresh, and Sanjiv Kumar. Sampled softmax with random fourier features. In *Proc. of NeurIPS*, 2019.

Aurko Roy, Mohammad Taghi Saffar, David Grangier, and Ashish Vaswani. Efficient content-based sparse attention with routing transformers, 2020.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proc. of ACL*, 2016.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In *Proc. of AAAI*, 2020.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers. In *Proc. of ACL*, 2019.

Yitong Sun. *Random Features Methods in Supervised Learning*. PhD thesis, The University of Michigan, 2019.

Yi Tay, Dara Bahri, Liu Yang, Don Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *Proc. of ICML*, 2020a.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey, 2020b.

Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *Proc. of EMNLP*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of NeurIPS*, 2017.

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

Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *Proc. of ICLR*, 2019.

Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite transformer with long-short range attention. In *Proc. of ICLR*, 2020.

Weiqiu You, Simeng Sun, and Mohit Iyyer. Hard-coded Gaussian attention for neural machine translation. In *Proc. of ACL*, 2020.

Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. In *Proc. of NeurIPS*, 2016.

Appendices

A Random Feature Attention in More Details

Algos. 1 and 2 describe causal and cross random feature attention's computation procedures.

Algorithm 1 Causal random feature attention.

```
procedure RFA-Causal( $\{\mathbf{q}_i\}_{i=1}^N$, $\{\mathbf{k}_i\}_{i=1}^N$, $\{\mathbf{v}_i\}_{i=1}^N$)
    ▷ $\mathbf{S}$ is a $D \times d$ matrix
    ▷ $\mathbf{z}$ is a $D$-dimensional vector
    $\mathbf{S}, \mathbf{z} \leftarrow \mathbf{0}, \mathbf{0}$
    for $i = 1$ to $N$ do
        $\widetilde{\mathbf{q}}_i, \widetilde{\mathbf{k}}_i \leftarrow \boldsymbol{\phi}(\mathbf{q}_i), \boldsymbol{\phi}(\mathbf{k}_i)$    ▷ Random feature maps
        $\mathbf{S} \leftarrow \mathbf{S} + \widetilde{\mathbf{k}}_i \otimes \mathbf{v}_i$
        $\mathbf{z} \leftarrow \mathbf{z} + \widetilde{\mathbf{k}}_i$
        $\mathbf{h}_i^\top \leftarrow \widetilde{\mathbf{q}}_i^\top \mathbf{S} / \widetilde{\mathbf{q}}_i \cdot \mathbf{z}$
    end for
    return $\{\mathbf{h}_i\}_{i=1}^N$
end procedure
```

Algorithm 2 Cross random feature attention.

```
procedure RAF-Cross( $\{\mathbf{q}_i\}_{i=1}^N$, $\{\mathbf{k}_i\}_{i=1}^M$, $\{\mathbf{v}_i\}_{i=1}^M$)
    ▷ $\mathbf{S}$ is a $D \times d$ matrix
    ▷ $\mathbf{z}$ is a $D$-dimensional vector
    $\mathbf{S}, \mathbf{z} \leftarrow \mathbf{0}, \mathbf{0}$
    for $i = 1$ to $M$ do
        $\widetilde{\mathbf{k}}_i \leftarrow \boldsymbol{\phi}(\mathbf{k}_i)$    ▷ Random feature map
        $\mathbf{S} \leftarrow \mathbf{S} + \widetilde{\mathbf{k}}_i \otimes \mathbf{v}_i^\top$
        $\mathbf{z} \leftarrow \mathbf{z} + \widetilde{\mathbf{k}}_i$
    end for
    for $i = 1$ to $N$ do
        $\widetilde{\mathbf{q}}_i \leftarrow \boldsymbol{\phi}(\mathbf{q}_i)$    ▷ Random feature map
        $\mathbf{h}_i^\top \leftarrow \widetilde{\mathbf{q}}_i^\top \mathbf{S} / \widetilde{\mathbf{q}}_i \cdot \mathbf{z}$
    end for
    return $\{\mathbf{h}_i\}_{i=1}^N$
end procedure
```

RAF without norm-1 constraints. §3.1 assumes that the queries and keys are unit vectors. This norm-1 constraint is *not* a must. Here we present a Rfa *without* imposing this constraint. Let $C(\mathbf{x}) = \exp(\|\mathbf{x}\|^2 / 2\sigma^2)$. From Eq. 4 we have $\operatorname{attn}(\mathbf{q}_t, \{\mathbf{k}_i\}, \{\mathbf{v}_i\}) =$

$$\begin{aligned}\sum_i \frac{\exp\left(\mathbf{q}_t \cdot \mathbf{k}_i / \sigma^2\right)}{\sum_j \exp\left(\mathbf{q}_t \cdot \mathbf{k}_j / \sigma^2\right)} \mathbf{v}_i^\top &\approx \sum_i \frac{C(\mathbf{q}_t)\, C(\mathbf{k}_i)\, \boldsymbol{\phi}\left(\mathbf{q}_t\right)^\top \boldsymbol{\phi}\left(\mathbf{k}_i\right) \mathbf{v}_i^\top}{\sum_j C(\mathbf{q}_t)\, C(\mathbf{k}_j)\, \boldsymbol{\phi}\left(\mathbf{q}_t\right) \cdot \boldsymbol{\phi}\left(\mathbf{k}_j\right)} \\ &= \frac{C(\mathbf{k}_i)\, \boldsymbol{\phi}\left(\mathbf{q}_t\right)^\top \sum_i \boldsymbol{\phi}\left(\mathbf{k}_i\right) \otimes \mathbf{v}_i}{\boldsymbol{\phi}\left(\mathbf{q}_t\right) \cdot \sum_j C(\mathbf{k}_j)\, \boldsymbol{\phi}\left(\mathbf{k}_j\right)}.\end{aligned} \tag{10}$$

The specific attention computation is similar to those in §3.1. In sum, lifting the norm-1 constraint brings an additional scalar term $C(\cdot)$.

Detailed complexity analysis. Table 3 considers a sequence-to-sequence model, and breaks down the comparisons to training (with teacher forcing; Williams & Zipser, 1989) and autoregressive decoding. Rfa has a lower space complexity, since it never explicitly populates the attention matrices.

As for time, RFA trains in linear time, and so does the softmax attention: in teacher-forcing training a standard transformer decoder parallelizes the attention computation across time steps. The trend of the time comparison differs during decoding: when only one output token is produced at a time, RFA decodes linearly in the output length, while softmax attention quadratically.

		Time Complexity			Space Complexity		
Setting	**Model**	**Encoder**	**Cross**	**Causal**	**Encoder**	**Cross**	**Causal**
Training w/ teacher forcing	softmax	$\mathcal{O}(M)$	$\mathcal{O}(M)$	$\mathcal{O}(N)$	$\mathcal{O}(M^2)$	$\mathcal{O}(MN)$	$\mathcal{O}(N^2)$
	RFA	$\mathcal{O}(M)$	$\mathcal{O}(M)$	$\mathcal{O}(N)$	$\mathcal{O}(M)$	$\mathcal{O}(M+N)$	$\mathcal{O}(N)$
Decoding	softmax	$\mathcal{O}(M)$	$\mathcal{O}(MN)$	$\mathcal{O}(N^2)$	$\mathcal{O}(M^2)$	$\mathcal{O}(MN)$	$\mathcal{O}(N^2)$
	RFA	$\mathcal{O}(M)$	$\mathcal{O}(M+N)$	$\mathcal{O}(N)$	$\mathcal{O}(M)$	$\mathcal{O}(M+N)$	$\mathcal{O}(N)$

Table 3: Time and space complexity comparisons between RFA and its softmax counterpart in a sequence-to-sequence attentive model. M and N denote the lengths of the source and target sequences respectively. Teacher forcing training (Williams & Zipser, 1989) and autoregressive decoding are assumed. Blue color indicates the cases where RFA asymptotically outperforms softmax attention.

Data	**Train**	**Dev.**	**Test**	**Vocab.**
WikiText-103	103M	218K	246K	268K
WMT14 EN-DE	4.5M	3K	3K	32K
WMT14 EN-FR	4.5M	3K	3K	32K
IWSLT14 DE-EN	160K	7K	7K	9K/7K

Table 4: Some statistics for the datasets. WikiText-103 split sizes are in number of tokens, while others are in number of instances.

B Experimental Details

Table 4 summarizes some statistics of the datasets used in our experiments. Our implementation is based on JAX.[15]

B.1 Language Modeling

We compare the models using two model size settings, summarized in Table 5. We use the fixed sinusoidal position embeddings by Vaswani et al. (2017). All models are trained for up to 150K gradient steps using the Adam optimizer (Kingma & Ba, 2015). No ℓ_2-regularization is used. We apply early stopping based on development set perplexity. All models are trained using 16 TPU v3 accelerators, and tested using a single TPU v2 accelerator.

B.2 Machine Translation

WMT14. We use the fixed sinusoidal position embeddings by Vaswani et al. (2017). For both EN-DE and EN-FR experiments, we train the models using the Adam (with $\beta_1 = 0.1$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$) optimizer for up to 350K gradient steps. We use batch size of 1,024 instances for EN-DE, while 4,096 for the much larger EN-FR dataset. We apply early stopping based on development set BLEU. Following standard practice, we average 10 most recent checkpoints. The learning rate follows that by Vaswani et al. (2017). No ℓ_2 regularization or gradient clipping is used. All models are trained using 16 TPU v3 accelerators, and tested using a single TPU v2 accelerator. Other hyperparameters are summarized in Table 6.

[15] `https://github.com/google/jax`.

Hyperprams.	**Small**	**Big**
# Layers	6	16
# Heads	8	16
Embedding Size	512	1024
Head Size	64	64
FFN Size	2048	4096
Batch Size	64	64
Learning Rate	$[1 \times 10^{-4}, 2.5 \times 10^{-4}, 5 \times 10^{-4}]$	
Warmup Steps	6000	6000
Gradient Clipping Norm	0.25	0.25
Dropout	[0.05, 0.1]	[0.2, 0.25, 0.3]
Random Feature Map Size	64	64

Table 5: Hyperparameters used in the language modeling experiments.

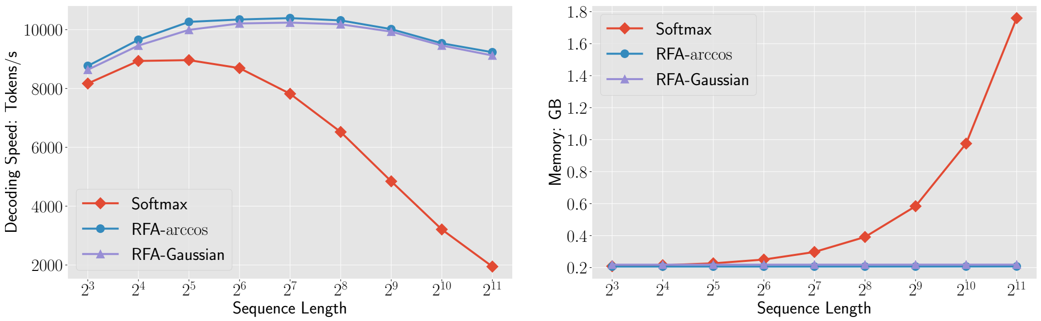
Hyperprams.	**WMT14**	**IWSLT14**
# Layers	6	6
# Heads	8	8
Embedding Size	512	512
Head Size	64	64
FFN Size	2048	2048
Warmup Steps	6000	4000
Dropout	0.1	0.3
Cross Attention Feature Map	128	128
Causal Attention Feature Map	64	64

Table 6: Hyperparameters used in the machine translation experiments.

C More Analysis Results

C.1 More Results on Decoding Speed and Memory Overhead

Figures 3 compare the RFA's *unconditional* decoding speed and memory against the softmax attention. The setting is the same as that in §5 except that here the models do not have an encoder. This experiment aims to simulate the applications such as sampling from a language model.



(a) Speed vs. lengths.

(b) Memory vs. lengths.

Figure 3: Unconditional decoding speed (left) and memory overhead (right) varying the output lengths. All models are tested on a single TPU v2 accelerator, with greedy decoding and batch size 16.

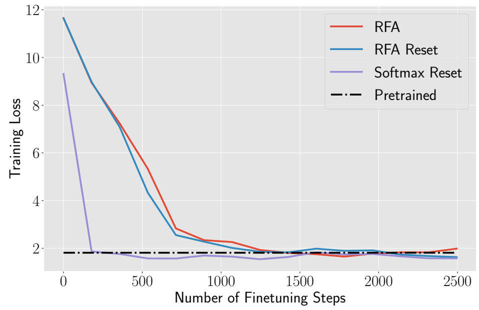


Figure 4: Finetuning an RFA-Gaussian model with its parameters initialized from a pretrained softmax-transformer. "Reset" indicates resetting the multihead attention parameters to randomly-initialized ones. The dashed line indicates the training loss of the pretrained model.

C.2 Train and Evaluate with Different Attention Functions

RFA achieves comparable performance to its softmax counterpart. Does this imply that it learns a good approximation to the softmax attention? To answer this question, we consider:

(i) an RFA-Gaussian model initialized from a pretrained softmax-transformer;
(ii) a softmax-transformer initialized from a pretrained an RFA-Gaussian model.

If RFA's good performance can be attributed to that it learns a good approximation to softmax, both, *without* finetunining, should perform similarly to the pretrained models. However, this is *not* the case on IWSLT14 DE-EN. Both pretrained models achieve more than 35.2 development set BLEU. In contrast, (i) and (ii) respectively get 2.3 and 1.1 BLEU *without* finetuning, hardly beating a randomly-initialized untrained model. This result aligns with the observation by Choromanski et al. (2020), and suggests that it is *not* the case that RFA performs well because it learns to imitate softmax attention's outputs.

C.3 Knowledge Transfer from Softmax Attention to RAF

We first supplement the observation in Appendix C.2 by finetuning (i) on the same pretraining data. Figure 4 plots the learning curves. It takes RFA roughly 1,500 steps to reach similar training loss to the pretrained model. As a baseline, "RFA Reset" resets the multihead attention parameters (i.e., those for query, key, value, and output projections) to randomly initialized ones. Its learning curve is similar to that of (i), suggesting that the pretrained multihead attention parameters are no more useful to RFA than randomly initialized ones. To further confirm this observation, "softmax Reset" resets the multihead attention parameters *without* changing the attention functions. It converges to the pretraining loss in less than 200 steps.

Takeaway. By the above results on IWSLT14, pretrained knowledge in a softmax transformer *cannot* be directly transferred to an RFA model. However, from Figure 4 and a much larger-scale experiment by Choromanski et al. (2020), we do observe that RFA can recover the pretraining loss, and the computation cost of finetuning is much less than training a model from scratch. This suggests some potential applications. For example, one might be able to initialize an RFA language model from a softmax transformer pretrained on large-scale data (e.g., GPT-3; Brown et al., 2020), and finetune it at a low cost. The outcome would be an RFA model retaining most of the pretraining knowledge, but is much faster and more memory-friendly to sample from. We leave such exploration to future work.