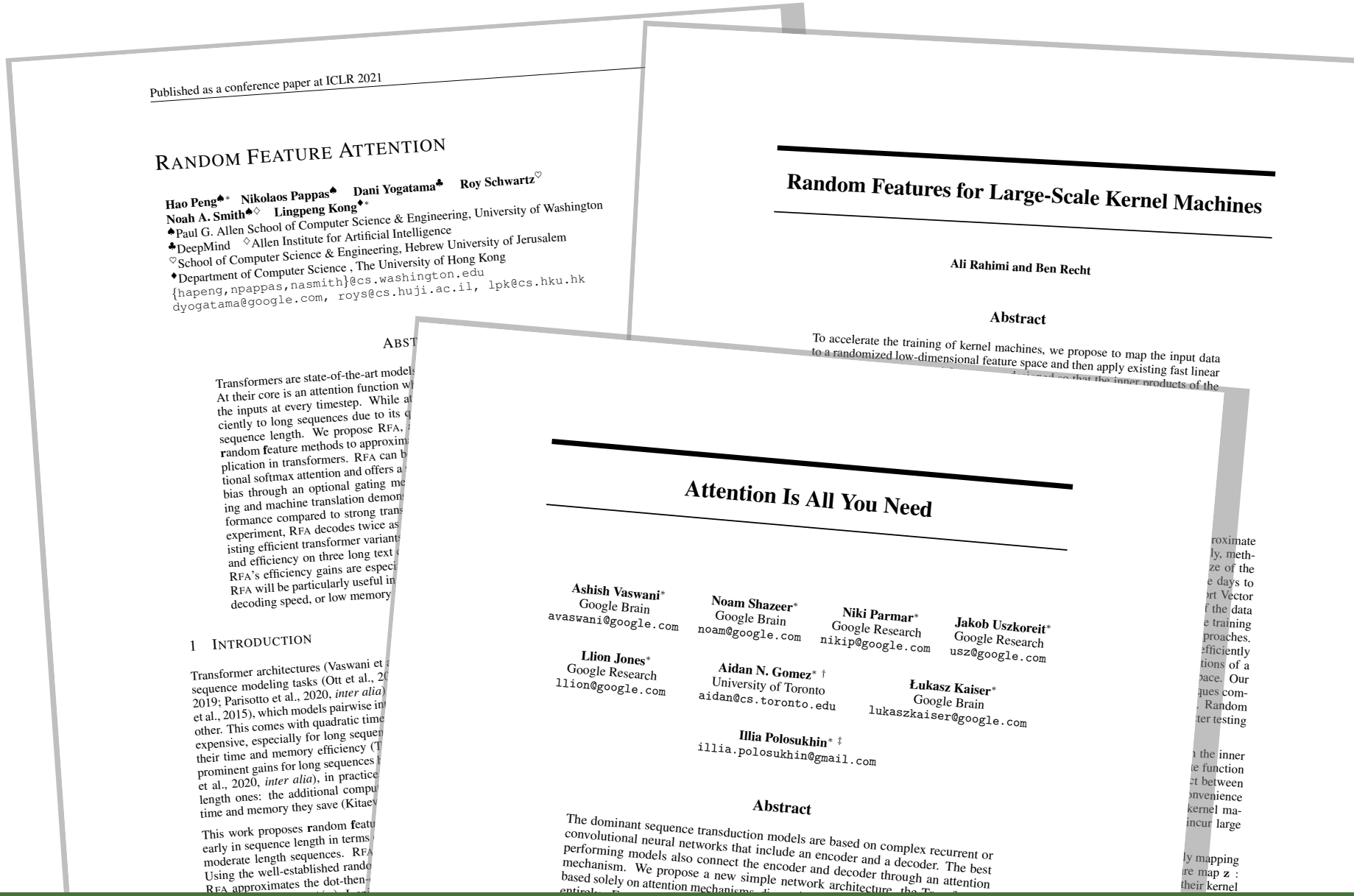
Table of Contents

1. Token Embedding
2. Time Complexity of Transformer Components
3. Model Scale: Key Dimensions
4. From Classical Analysis ...
5. ... to Modern Machine Learning
6. Results
7. Limitations and Trade-offs
8. Alternative Approaches

Main References



Published as a conference paper at ICLR 2021

RANDOM FEATURE ATTENTION

Hao Peng[♠*] **Nikolaos Pappas**[♠] **Dani Yogatama**[♣] **Roy Schwartz**[♡]
Noah A. Smith[♠◇] **Lingpeng Kong**[♦*]
[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♣]DeepMind [◇]Allen Institute for Artificial Intelligence
[♡]School of Computer Science & Engineering, Hebrew University of Jerusalem
[♦]Department of Computer Science , The University of Hong Kong
{hapeng,npappas,nasmith}@cs.washington.edu
dyogatama@google.com, roys@cs.huji.ac.il, lpk@cs.hku.hk

ABST

Transformers are state-of-the-art models
At their core is an attention function w
the inputs at every timestep. While at
ciently to long sequences due to its q
sequence length. We propose RFA,
random feature methods to approxim
plication in transformers. RFA can b
tional softmax attention and offers a
bias through an optional gating me
ing and machine translation demon
formance compared to strong trans
experiment, RFA decodes twice as
isting efficient transformer variant
and efficiency on three long text
RFA's efficiency gains are especi
RFA will be particularly useful in
decoding speed, or low memory

1 INTRODUCTION

Transformer architectures (Vaswani et a
sequence modeling tasks (Ott et al., 20
2019; Parisotto et al., 2020, *inter alia*)
et al., 2015), which models pairwise in
other. This comes with quadratic time
expensive, especially for long sequen
their time and memory efficiency (T
prominent gains for long sequences
et al., 2020, *inter alia*), in practice
length ones: the additional compu
time and memory they save (Kitaev

This work proposes random featu
early in sequence length in terms
moderate length sequences. RFA
Using the well-established rando
RFA approximates the dot-then-

Random Features for Large-Scale Kernel Machines

Ali Rahimi and Ben Recht

Abstract

To accelerate the training of kernel machines, we propose to map the input data to a randomized low-dimensional feature space and then apply existing fast linear

roximate
ly, meth-
ze of the
e days to
rt Vector
f the data
e training
proaches.
fficiently
tions of a
pace. Our
ques com-
. Random
ter testing

the inner
e function
t between
nvenience
kernel ma-
incur large

ly mapping
re map $\mathbf{z}$:
their kernel

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[* †]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

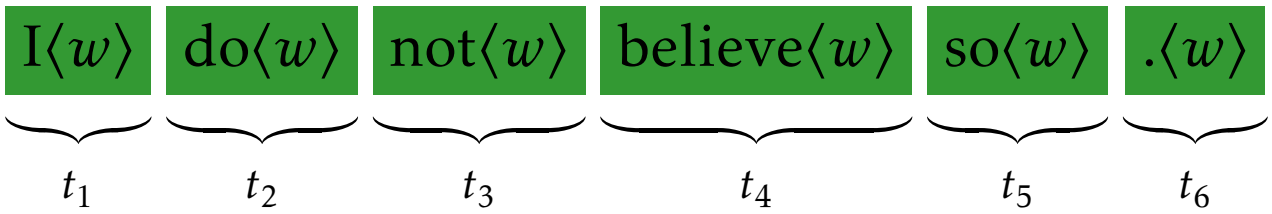
Illia Polosukhin[* ‡]
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the T
based solely on attention mechanisms, di
entirely

Tokenization

516 247$^{\text{th}}$ entry of WMT14 En→De:
Tokenized (EN):



Tokenized (DE):

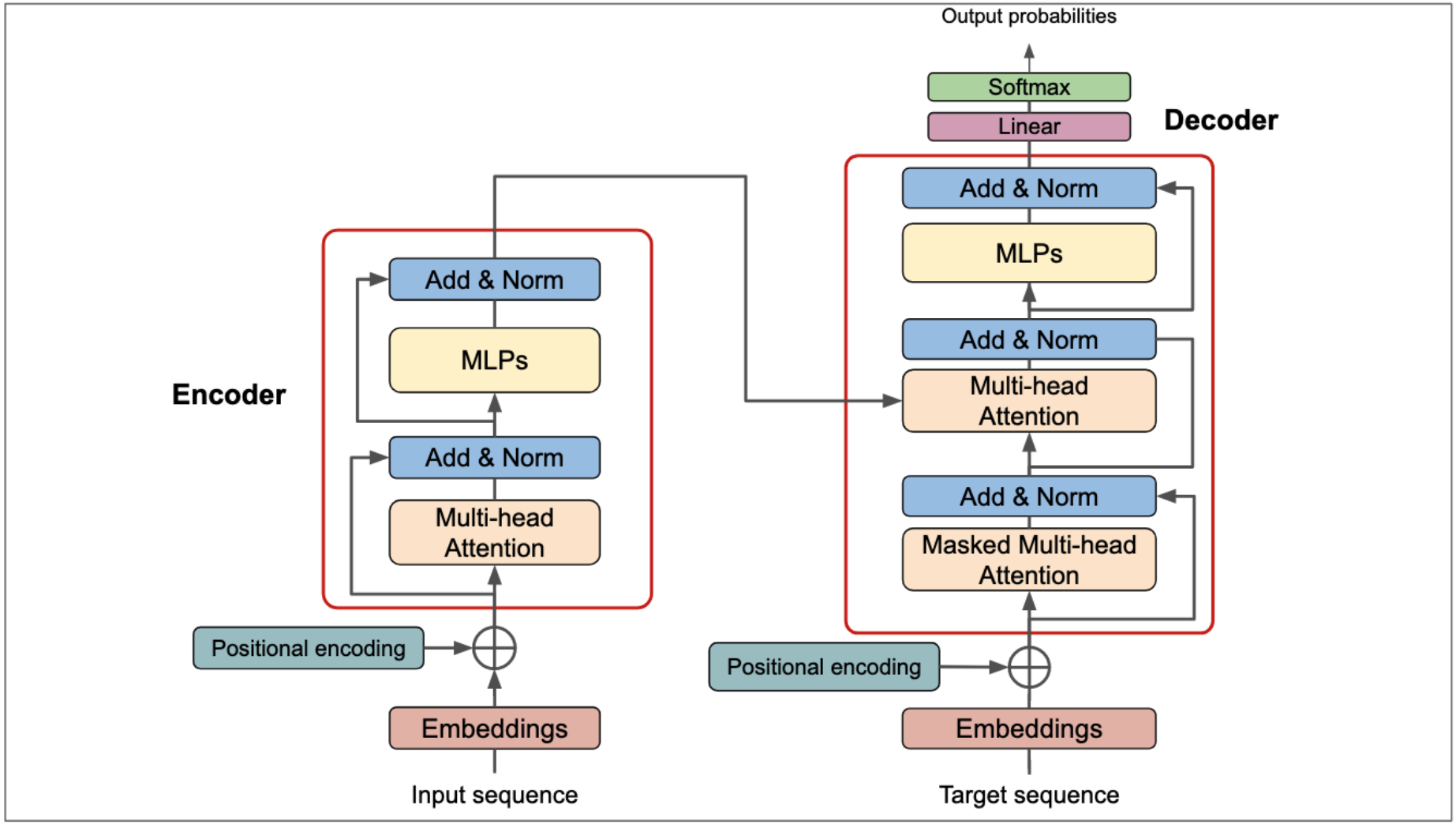
Ich⟨w⟩ glaube⟨w⟩ ,⟨w⟩ nein⟨w⟩ .⟨w⟩

Embedding

- # Vocabulary: $V \approx 50\,000$
- **Embedding dimension**: $d_m \approx 512 - 12\,288$
- Embedding dictionary: $D \in \mathbb{R}^{V \times d_m}$
- **Context length**: $n \in \mathbb{N}$
- Tokens: $t_1, \ldots, t_n$
- Embedded tokens: $t_i \longrightarrow \text{Emb}[t_i] \in \mathbb{R}^{d_m}, \quad i = 1, \ldots, n$

$$t_i = \text{“cat”} \quad \longrightarrow \quad \text{Emb}[\text{“cat”}] = \begin{bmatrix} 0.4 \\ 0.3 \\ \vdots \end{bmatrix} \in \mathbb{R}^{d_m}$$

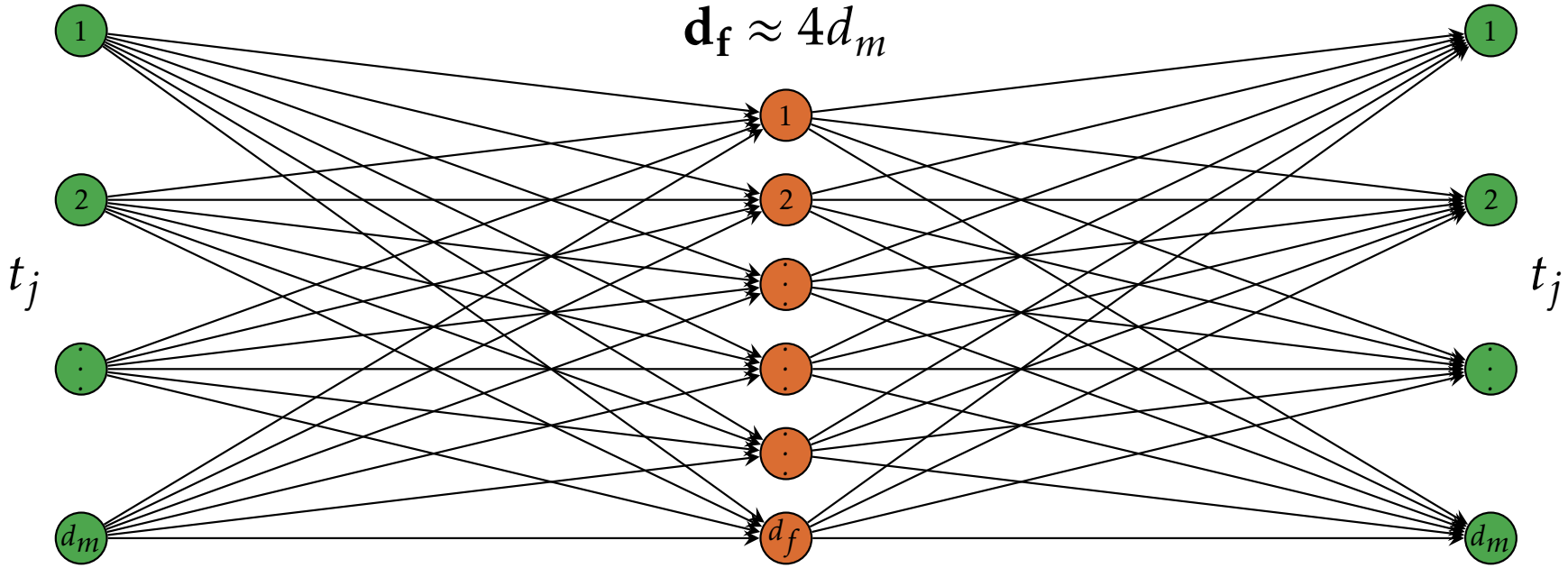
Transformer Architecture[1] [Vas+17]



[1] https://deeprevision.github.io/posts/001-transformer/

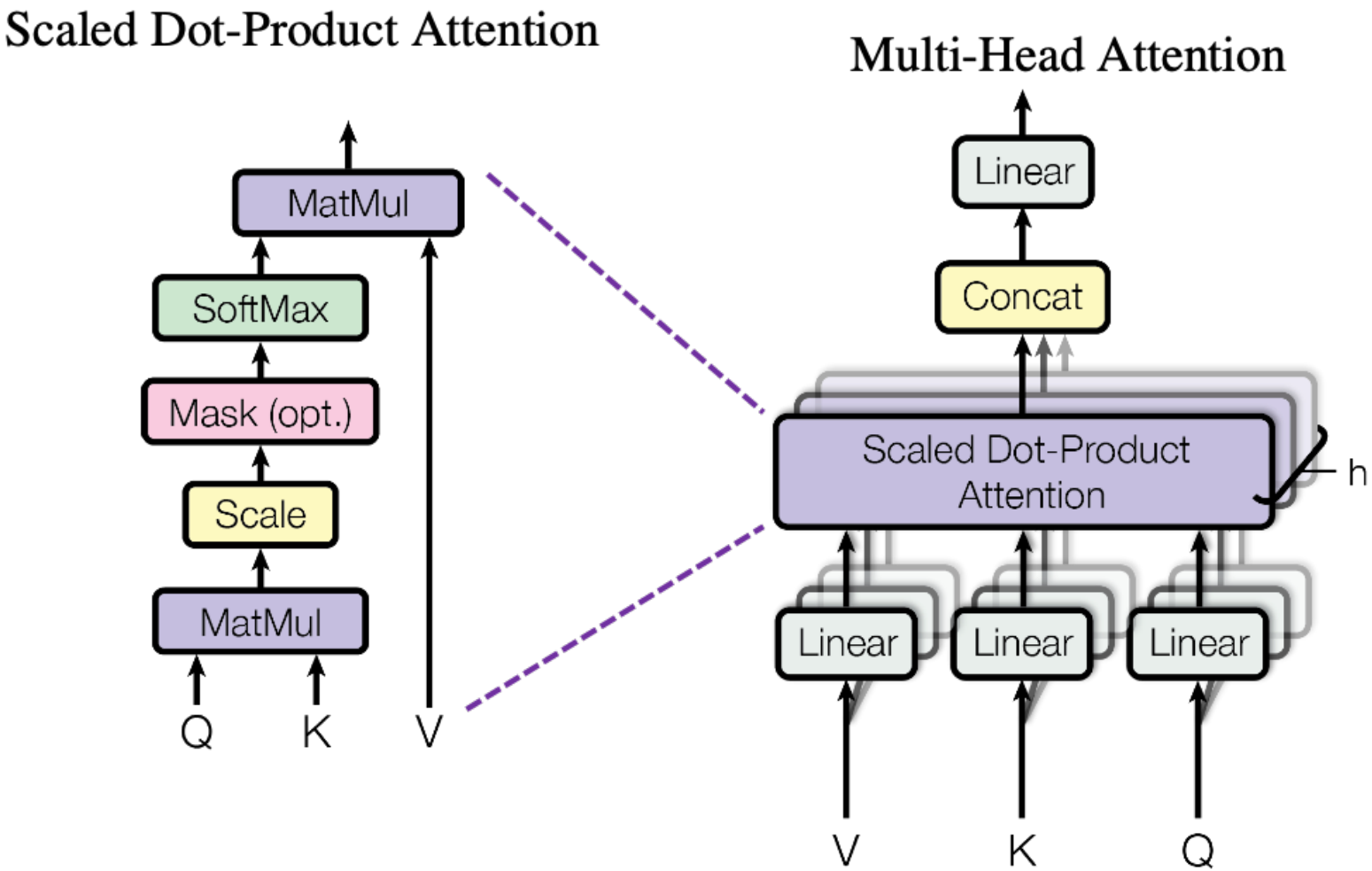
Time Complexity: The Feed Forward Network (MLP)

$W \in \mathbb{R}^{d_f \times d_m}, b \in \mathbb{R}^{d_f}$ $\quad$ $W \in \mathbb{R}^{d_m \times d_f}, b \in \mathbb{R}^{d_m}$



$$\mathcal{O}(n(\underbrace{d_f d_m + d_f}_{\text{layer 1}} + \underbrace{d_f}_{\text{ReLU}} + \underbrace{d_m d_f + d_m}_{\text{layer 2}})) = \mathcal{O}(nd_m^2)$$

The Attention Mechanism [Vas+17]



Time Complexity: The Attention Mechanism

- For **heads** $h = 1, \dots, H \in \mathbb{N}$ and **Attention dimension** $d_H := d_m / H$
- $Q_h, K_h, V_h \in \mathbb{R}^{d_H \times d_m}$
- $q_j = Q_h t_j, \quad k_j = K_h t_j, \quad v_j = V_h t_j \in \mathbb{R}^{d_H}$ for token $t_1, \dots, t_n \in \mathbb{R}^{d_m}$ in $\mathcal{O}(n d_m^2)$.
- $$\mathrm{Atn}_{k,h} = \sum_{j=1}^{n} \frac{\mathrm{e}^{\langle q_k, k_j \rangle}}{\sum_{l=1}^{n} \mathrm{e}^{\langle q_k, k_l \rangle}} v_j \in \mathbb{R}^{d_H}, \quad k = 1, \dots, n \qquad (1)$$
- $t_j \longrightarrow \mathrm{Atn}_{j,h} \in \mathbb{R}^{d_m}, j = 1, \dots, n$ in $\mathcal{O}(nH(nd_H)d_H) = \mathcal{O}(n^2 d_m d_H)$.

Transformer Stats

- **Time Complexity:** $\mathcal{O}(\underbrace{n^2 d_m d_H + n d_m^2}_{\text{Attention}} + \underbrace{n d_m^2}_{\text{MLP}})$

Parameter		[Vas+17]	[Bro+20]
Vocabulary	V	37 000	50 257
Embedding dimension	d_m	512	12 288
Context window	n_{ctx}	(512)	2 048
MLP dimension	d_f	2 048	49 152
Heads	H	8	96
Attention dimension	d_H	64	128
Attention–MLP blocks	N	6	96
FLOPS for training	flops	(6E+18)	3.14E+23
Peta Flop days	PF-days	(7E-02)	3.64E+03

Table: Comparison of Transformer sizes.

From Classical Analysis …

- Positive definite Kernels/Functions

$$K \in \mathrm{PD}(\mathbb{R}^d) \Leftrightarrow \sum_{j,k=1}^{n} \alpha_j K(x_j, x_k)\alpha_k \geq 0, \forall n \in \mathbb{N}, \alpha \in \mathbb{R}^n, x \in \mathbb{R}^{n \times d}$$

- [Mer09] Mercers's Theorem

$$K \in \mathrm{PD}(\mathbb{R}^d) \Rightarrow K(x,y) = \langle \Phi(x), \Phi(y) \rangle_{\ell^2}$$

- [Boc33] Bochner's Theorem $K(x,y) = f(x-y)$

$$\mathcal{F} : \mathcal{P}(\mathbb{R}^d) \leftrightarrow \widetilde{\mathrm{PD}}(\mathbb{R}^d), \quad f(x) = \mathcal{F}[\mu](x) := \int_{\mathbb{R}^d} \mathrm{e}^{\mathrm{i}\langle x,y \rangle} \mathrm{d}\mu(y).$$

- [CV95] Support Vector Machines $K(x,y) = \mathrm{e}^{-\frac{1}{2}\|x-y\|^2}$

$$\sum_{j=1}^{n} \alpha_j K(x_j, y_l), \quad l = 1, \dots n \quad \mathcal{O}(n^2)$$

… to Modern Machine Learning

- [RR07] Random Fourier Features $f(x-y) = \mathrm{e}^{-\frac{1}{2}\|x-y\|_2^2}$

$$f(x) = \mathcal{F}[\mu](x) = \int_{\mathbb{R}^d} \mathrm{e}^{\mathrm{i}\langle x,y\rangle} \mathrm{d}\mu(y) = \mathbb{E}_{\xi\sim\mu}[\mathrm{e}^{\mathrm{i}\langle \xi,x\rangle}] \approx \sum_{k=1}^{P} \frac{\mathrm{e}^{\mathrm{i}\langle \xi_k,x\rangle}}{P}.$$

$$\sum_{j=1}^{n} \alpha_j K(x_j, y_l) \approx \sum_{j=1}^{n} \frac{\alpha_j}{P} \sum_{k=1}^{P} \mathrm{e}^{\mathrm{i}\langle \xi_k, x_j - y_l\rangle} = \sum_{k=1}^{P} \mathrm{e}^{\mathrm{i}\langle \xi_k, y_l\rangle} \sum_{j=1}^{n} \frac{\alpha_j}{P} \mathrm{e}^{\mathrm{i}\langle \xi_k, x_j\rangle}$$

$\Rightarrow$ Time Complexity $\mathcal{O}(Pnd)$ for $P \ll n$ good.

$\Rightarrow$ # Features $P \approx d$ depends on $\mathbb{R}^d$ but not on $n \sim \mathcal{O}(nd^2)$.

$\Rightarrow$ Generalizable to arbitrary positive definite functions

Application in the Attention Mechanism

- [Vas+17] Attention Is All You Need
- [Kat+20; Pen+21; Cho+21]

$$\text{Atn}_{k,h} = \sum_{j=1}^{n} \frac{e^{\langle q_k, k_j \rangle}}{\sum_{l=1}^{n} e^{\langle q_k, k_l \rangle}} v_j \in \mathbb{R}^{d_H}, \quad k = 1, \dots, n$$

$\Rightarrow$

$$e^{\langle x,y \rangle} = e^{-\frac{1}{2}\|x\|^2 - \frac{1}{2}\|y\|^2 - \frac{1}{2}\|x-y\|^2} = e^{-\frac{1}{2}\|x\|^2} e^{-\frac{1}{2}\|y\|^2} e^{-\frac{1}{2}\|x-y\|^2}$$

$\Rightarrow$

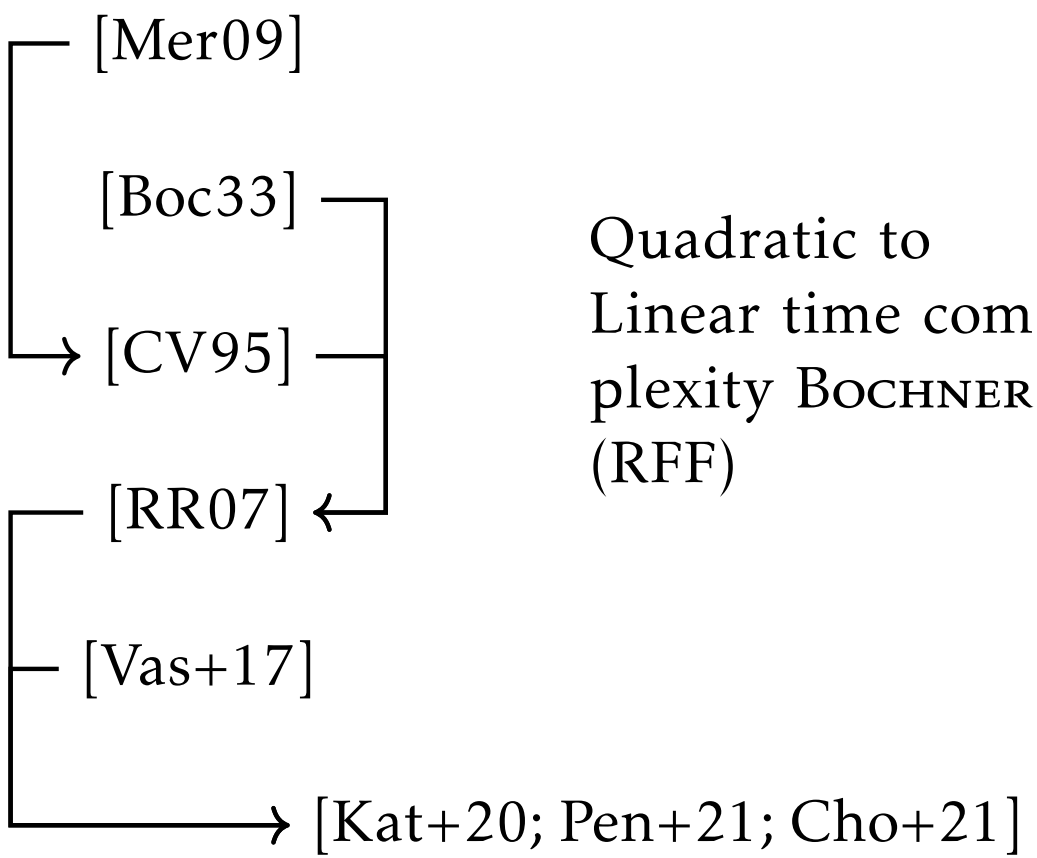
$$\text{Atn}_{k,h} = \sum_{j=1}^{n} \frac{e^{\frac{1}{2}\|q_k - k_j\|^2}}{\sum_{l=1}^{n} e^{-\frac{1}{2}\|q_k - k_l\|} e^{-\frac{1}{2}\|k_l\|^2}} v_j e^{-\frac{1}{2}\|k_j\|^2} = \frac{\sum_{j=1}^{n} \alpha_j K(q_k, k_j)}{\sum_{l=1}^{n} \beta_l K(q_k, k_l)}$$

- Runtime: Attention
  $\mathcal{O}(nd_m^2 + H(Pnd_H)d_H) \to \mathcal{O}(nd_m^2 + nd_H^2 d_m)$.

A Timeline: From Classical Analysis to Modern Machine Learning

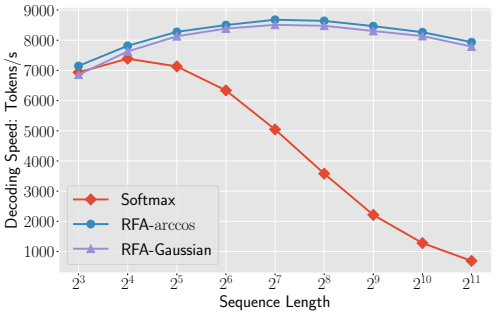
Nonlinear classification via SVMs Mercer (Kernel trick)

RFF applied to Attention mechanism

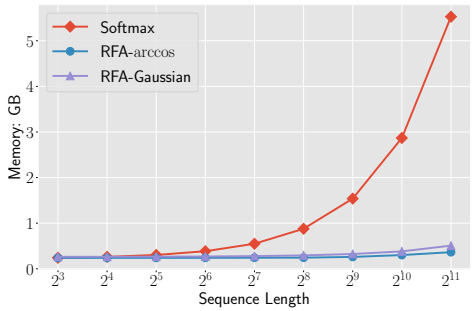


Quadratic to Linear time complexity Bochner (RFF)

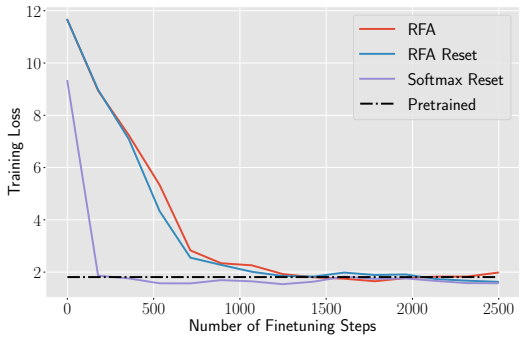
Results: Random Feature Attention [Pen+21]



(a) Speed vs. lengths.



(b) Memory vs. lengths.



- **Drop-in replacement** for standard attention
- **12× faster decoding** with **<10% memory** usage for 2048-token outputs
- **Not** an approximation of Softmax
- Compatible with **pretrained models**

Results: Rethinking Attention with Performers [Cho+21]

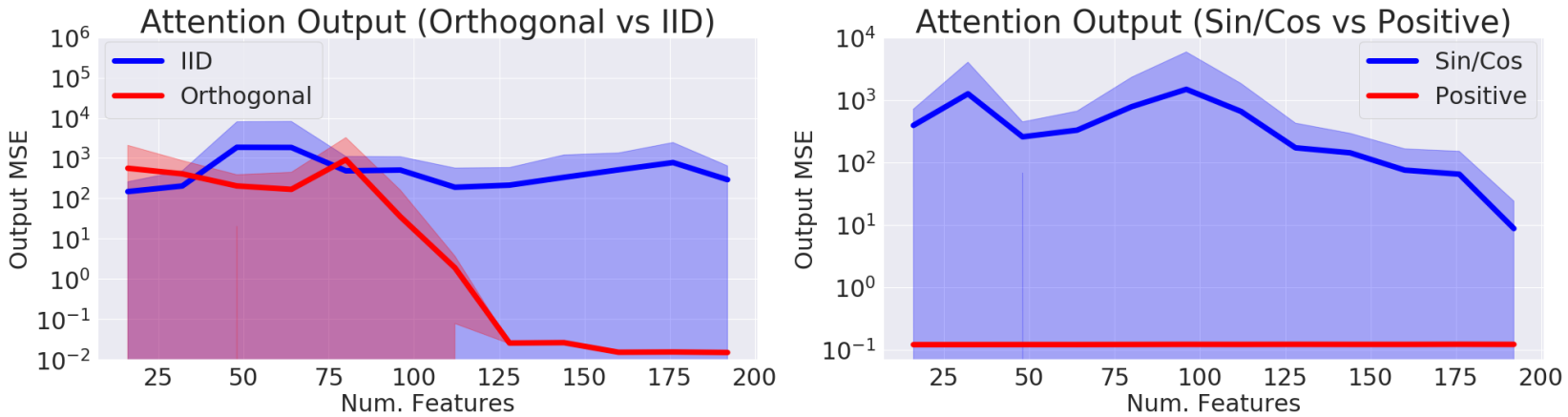


Figure 4: MSE of the approximation output when comparing Orthogonal vs IID features and trigonometric sin/cos vs positive features. We took $L = 4096, d = 16$, and varied the number of random samples m. Standard deviations shown across 15 samples of appropriately normalized random matrix input data.

Limitations and Trade-offs

- Complexity improves when feature size $P \approx d_H$:

$$\mathcal{O}(nd_m^2 + H(Pnd_H)d_H) \;\rightarrow\; \mathcal{O}(nd_H^2 d_m).$$

- If $n \approx d_m$: overall cost dominated by MLP,

$$\mathcal{O}(n^2 d_m d_H + nd_m^2) \;\rightarrow\; \mathcal{O}(d_m^3).$$

$\Rightarrow$ Best suited for **long-context, low-dimension** Transformers.

- Training may take more steps, but inference is significantly faster.
- Further gains possible via **orthogonal sampling** and **positive feature maps**.
- This method **does not approximate softmax**.

Alternative Approaches

- **Windowing** — restrict attention to fixed local *windows of tokens* instead of the full sequence.
- **Recurrence** — extend context by *reusing cached hidden states* from previous segments.
- **Clustering** — reduce complexity by *grouping tokens with locality-sensitive hashing* and attending only within clusters.
- **Linformer** — approximate attention with a *low-rank projection of keys and values*, yielding linear complexity.
- **FlashAttention** — speed up training with *GPU-efficient blockwise computation of exact softmax attention*.

Thank you for your (random feature) Attention!



Attention **Optimized** to its **Prime**[2]

[2] https://dropinblog.net/cdn-cgi/image/fit=scale-down,width=700/34260623/files/featured/optimus-prime-1.webp

References I

[Boc33] Salomon Bochner. "Monotone funktionen, stieltjessche integrale und harmonische analyse". In: *Mathematische Annalen* 108.1 (1933), pp. 378–410.

[Bro+20] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[Cho+21] Krzysztof Marcin Choromanski et al. "Rethinking Attention with Performers". In: *International Conference on Learning Representations*. 2021.

[CV95] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[Kat+20] Angelos Katharopoulos et al. "Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention". In: *International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 5156–5165.

[Mer09] James Mercer. "Xvi. functions of positive and negative type, and their connection the theory of integral equations". In: *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character* 209.441-458 (1909), pp. 415–446.

[Pen+21] Hao Peng et al. "Random Feature Attention". In: *International Conference on Learning Representations*. 2021.

[RR07] Ali Rahimi and Benjamin Recht. "Random features for large-scale kernel machines". In: *Advances in neural information processing systems* 20 (2007).

[Vas+17] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

Future ideas

1. Instead of RFF use slicing
2. Apply to other kernels then Gaussians / learn coefficients of f
3. How to reduce Time Complexity of MLP
4. Can the Attention parameters be swapped in to transformers?
5. Improve slicing directions? $\rightarrow$ QMC/orthogonal
6. Better to have positive coefficients?
7. What are the really efficient methods for large attention windows? Can they be combined?

Slicing summation

- Idea: Construct even $f : \mathbb{R} \to \mathbb{R}$ s.t.

$$F(\|x\|) = \frac{1}{|\mathbb{S}^{d-1}|} \int_{\mathbb{S}^{d-1}} f(|\langle x, \xi \rangle|) \mathrm{d}\xi \approx \frac{1}{P} \sum_{p=1}^{P} f(|\langle x, \xi_p \rangle|)$$

- Insert into the sum $s_m = \sum_{n=1}^{N} w_n F(\|x_n - y_m\|)$

$$s_m = \sum_{n=1}^{N} w_n \frac{1}{|\mathbb{S}^{d-1}|} \int_{\mathbb{S}^{d-1}} f(|\langle x_n - y_m, \xi \rangle|) \mathrm{d}\xi$$

- Choose points $\xi_1, \ldots, \xi_P \in \mathbb{S}^{d-1}$

$$s_m \approx \frac{1}{P} \sum_{p=1}^{P} \underbrace{\sum_{n=1}^{N} w_n f(|\langle x_n, \xi_p \rangle - \langle y_m, \xi_p \rangle|)}_{\text{1D kernel sum}}$$

- P one-dimensional sums $\rightsquigarrow \mathcal{O}(P(N + M + N_{\mathrm{ft}} \log N_{\mathrm{ft}}))$

Fast Fourier Summation

- Fourier expansion of the kernel with degree $N_{\mathrm{ft}} \in \mathbb{N}$

$$f(|x|) \approx \sum_{k=-N_{\mathrm{ft}}}^{N_{\mathrm{ft}}} c_k \mathrm{e}^{2\pi \mathrm{i}\langle x,k\rangle}$$

- For $x_1,\dots,x_N,y_1,\dots,y_M \in \mathbb{R}$

$$\underbrace{\sum_{n=1}^{N} w_n f(|x_n - y_m|)}_{s_m} \approx \sum_{n=1}^{N} w_n \sum_{k=-N_{\mathrm{ft}}}^{N_{\mathrm{ft}}} c_k \mathrm{e}^{2\pi \mathrm{i} k(x_n - y_m)} = \sum_{k=-N_{\mathrm{ft}}}^{N_{\mathrm{ft}}} c_k \mathrm{e}^{-2\pi \mathrm{i} y_m k} \underbrace{\sum_{n=1}^{N} w_n \mathrm{e}^{2\pi \mathrm{i} x_n k}}_{=:\hat{w}_k}$$

1. Comp. inner sum $\hat{w}_k$ via NFFT of $w \rightsquigarrow \mathcal{O}(N + N_{\mathrm{ft}} \log N_{\mathrm{ft}})$
2. Comp. outer sum s_m via NFFT of $c_k \hat{w}_k \rightsquigarrow \mathcal{O}(M + N_{\mathrm{ft}} \log N_{\mathrm{ft}})$
3. Total complexity: $\mathcal{O}(N + M + N_{\mathrm{ft}} \log N_{\mathrm{ft}})$