TaylorShift:

Shifting the Complexity of Self-Attention from Squared to Linear (and Back) using Taylor-Softmax

ICPR 2024 - Kolkata, India

Deutsches Forschungszentrum für Künstliche Intelligenz
German Research Center for Artificial Intelligence

RPTU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

Tobias Christian Nauen

Sebastian Palacio

Andreas Dengel

dfki ai RPTU

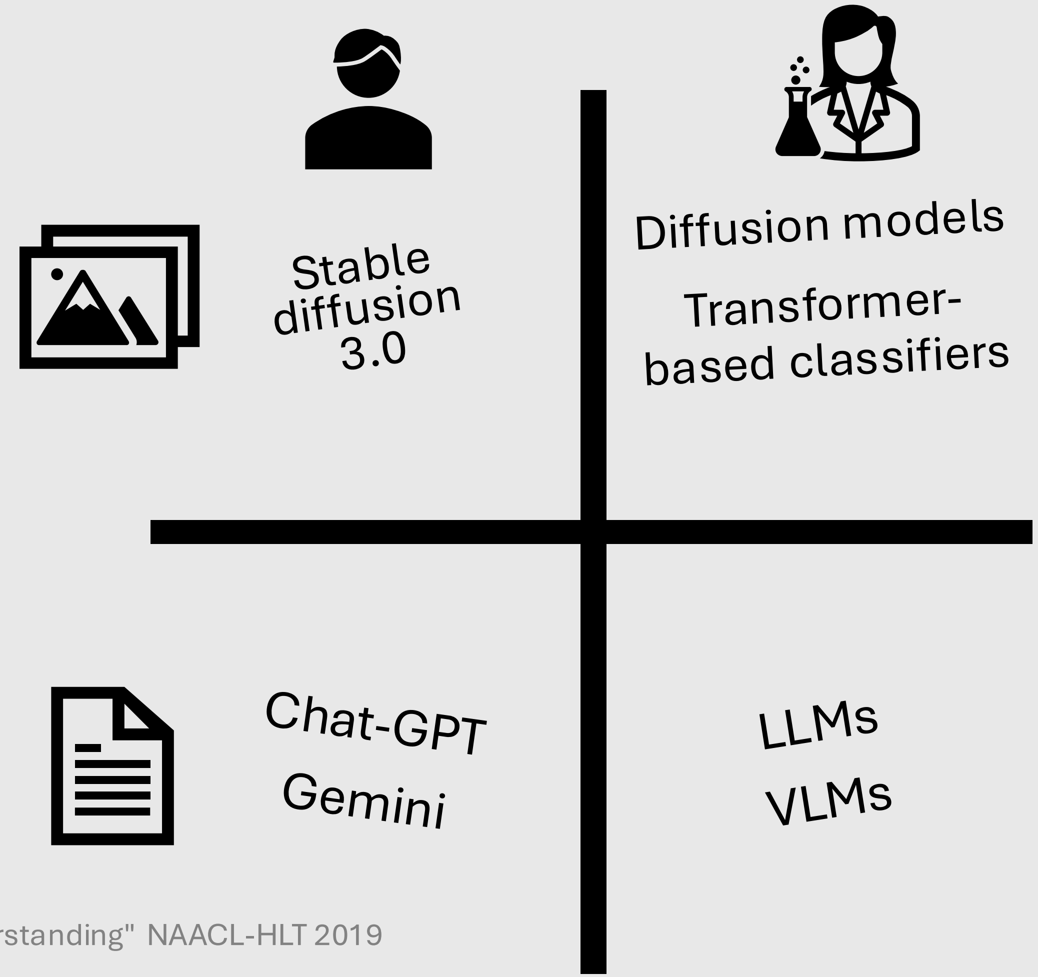
Modern AI is largely attention-based.

Introduction of Transformer models by Vaswani et al. [1]

BERT [2]

GPT-2 [3]

ViT [4]

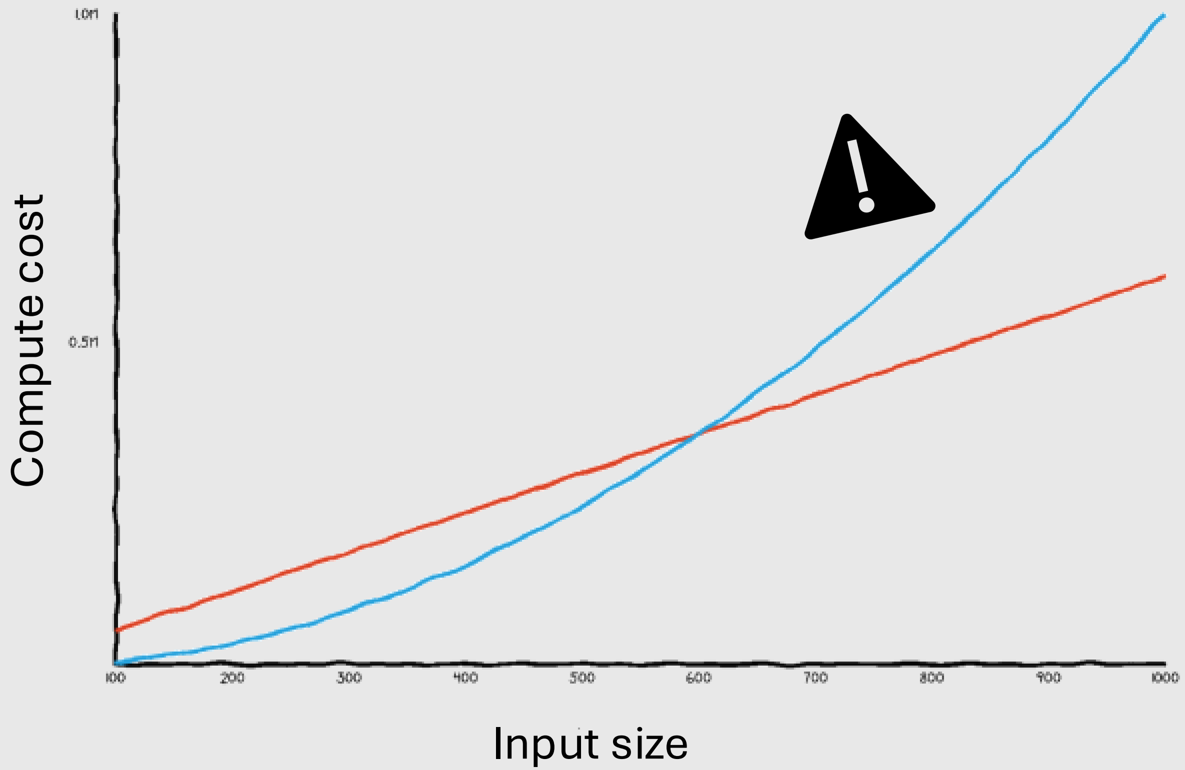


[1] Vaswani et al. "Attention Is All You Need" NeurIPS 2017
[2] Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" NAACL-HLT 2019
[3] Radford et al. "Language Models are Unsupervised Multitask Learners" 2019
[4] Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" ICLR 2021

Quadratic complexity prevents us from utilizing very long inputs.

Long inputs allow for:

- More context information
- (Updated) World knowledge

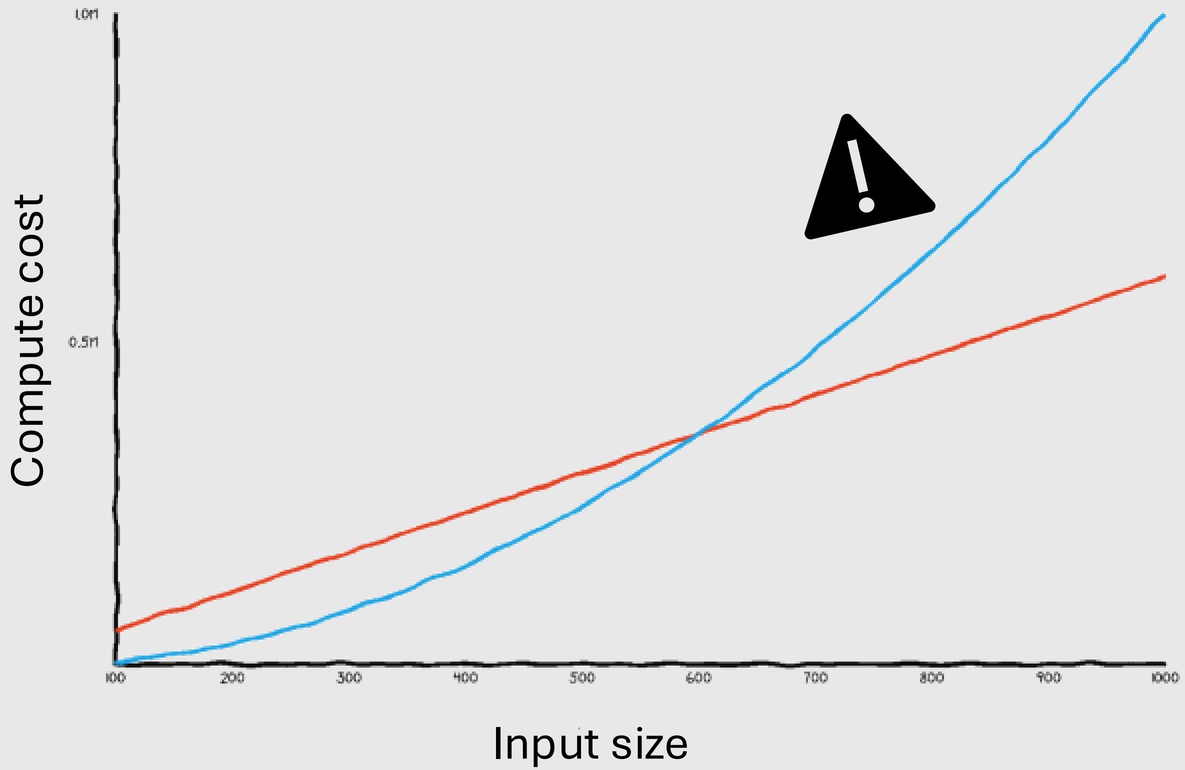


Quadratic complexity prevents us from utilizing very long inputs.

Long inputs allow for:

- More context information
- (Updated) World knowledge

- High-resolution images
- More localized information



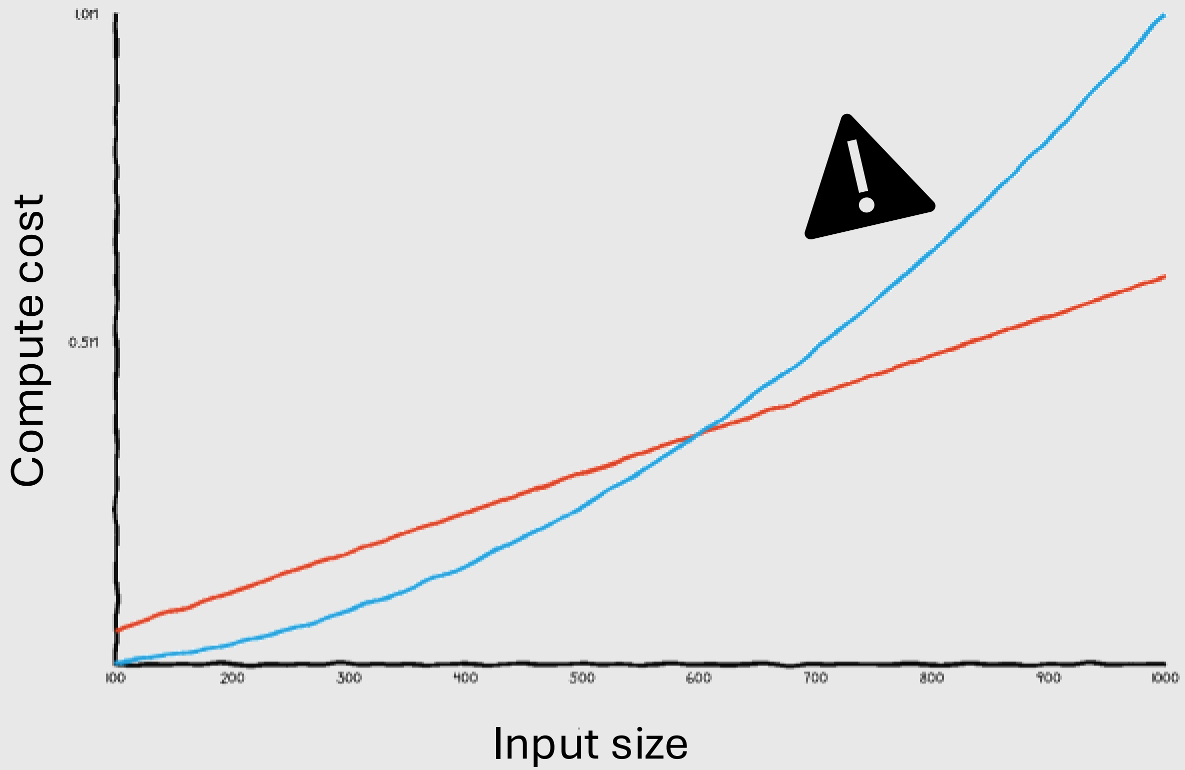
Quadratic complexity prevents us from utilizing very long inputs.

Long inputs allow for:

- More context information
- (Updated) World knowledge

- High-resolution images
- More localized information

- Multiple modalities
- Concatenation of information



TaylorShift is attention using the Taylor series of the exponential.

	Attention	**Direct-TaylorShift**	
Activation	$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}$	$\approx \sum_{n=0}^{k} \frac{x^n}{n!}$	$k = 2$

TaylorShift is attention using the Taylor series of the exponential.

	Attention	**Direct-TaylorShift**
Activation	$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}$	$\approx \sum_{n=0}^{k} \frac{x^n}{n!}$ $\quad k = 2$
Interaction Function	$\text{softmax}(x) = \frac{\exp(x)}{\|\| \exp(x) \|\|_1}$	$\text{T-SM}^{(k)}(x) = \frac{\sum_{n=0}^{k} \frac{x^n}{n!}}{\left\|\left\| \sum_{n=0}^{k} \frac{x^n}{n!} \right\|\right\|_1}$

TaylorShift is attention using the Taylor series of the exponential.

	Attention	Direct-TaylorShift
Activation	$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}$	$\approx \sum_{n=0}^{k} \frac{x^n}{n!}$ $\quad k = 2$
Interaction Function	$\mathrm{softmax}(x) = \frac{\exp(x)}{\lvert\lvert \exp(x) \rvert\rvert_1}$	$\text{T-SM}^{(k)}(x) = \frac{\sum_{n=0}^{k} \frac{x^n}{n!}}{\left\lvert\left\lvert \sum_{n=0}^{k} \frac{x^n}{n!} \right\rvert\right\rvert_1}$
Attention Calculation	$Y = \mathrm{softmax}(QK^\top)V$	$Y = \text{T-SM}^{(k)}(QK^\top)V$

Direct-TaylorShift

TaylorShift is attention using the Taylor series of the exponential.

dfki ai
RPTU

	Attention	**Direct-TaylorShift**
Activation	$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}$	$\approx \sum_{n=0}^{k} \frac{x^n}{n!}$ $\quad k = 2$
Interaction Function	$\mathrm{softmax}(x) = \frac{\exp(x)}{\|\| \exp(x) \|\|_1}$	$\text{T-SM}^{(k)}(x) = \frac{\sum_{n=0}^{k} \frac{x^n}{n!}}{\left\|\left\| \sum_{n=0}^{k} \frac{x^n}{n!} \right\|\right\|_1}$
Attention Calculation	$Y = \mathrm{softmax}(QK^\top)V$	$Y = \text{T-SM}^{(k)}(QK^\top)V$ Direct-TaylorShift
Complexity $Q, K, V \in \mathbb{R}^{N \times d}$	$\mathcal{O}(N^2 d)$	$\mathcal{O}(N^2 d)$

We can reorder calculations for efficient calculation.

$$\text{T-SM}(QK^\top)V \;=\; \ell_1 - \text{norm}\left[\sum_{n=0}^{2} \frac{(QK^\top)^n}{n!}\right] V$$

1. Matrix multiplication (QK)
2. Activation function
3. Normalization
4. Matrix multiplication (KV) $\mathcal{O}(N^2 d)$

We can reorder calculations for efficient calculation.

$$\text{T-SM}(QK^\top)V \;=\; \ell_1 - \text{norm}\left[\sum_{n=0}^{2}\frac{(QK^\top)^n}{n!}\right]V \;=\; \text{norm}\left[\sum_{n=0}^{2}\frac{(QK^\top)^n}{n!}V \circ 1\right]$$

1. Matrix multiplication (QK)
2. Activation function
3. Normalization
4. Matrix multiplication (KV) $\mathcal{O}(N^2 d)$

We can reorder calculations for efficient calculation.

$$\text{T-SM}(QK^\top)V = \ell_1 - \text{norm}\left[\sum_{n=0}^{2} \frac{(QK^\top)^n}{n!}\right]V = \text{norm}\left[\sum_{n=0}^{2} \frac{(QK^\top)^n}{n!} V \circ 1\right]$$

$$= \text{norm}\left[\frac{1}{2}(QK^\top)^2(V \circ 1) + QK^\top(V \circ 1) + \sum_{\text{cols}}(V \circ 1)\right]$$

1. Matrix multiplication (QK)
2. Activation function
3. Normalization
4. Matrix multiplication (KV) $\mathcal{O}(N^2 d)$

We can reorder calculations for efficient calculation.

$$\text{T-SM}(QK^\top)V \;=\; \ell_1 - \text{norm}\left[\sum_{n=0}^{2} \frac{(QK^\top)^n}{n!}\right] V \;=\; \text{norm}\left[\sum_{n=0}^{2} \frac{(QK^\top)^n}{n!} V \circ 1\right]$$

$$= \text{norm}\left[\frac{1}{2}(QK^\top)^2(V \circ 1) + QK^\top(V \circ 1) + \sum_{\text{cols}}(V \circ 1)\right]$$

$$= \text{norm}\left[\frac{1}{2} Q^{\boxtimes 2}(K^{\boxtimes 2})^\top(V \circ 1) + QK(V \circ 1) + \sum_{\text{cols}}(V \circ 1)\right]$$

Efficient-TaylorShift

1. Matrix multiplication (QK)
2. Activation function
3. Normalization
4. Matrix multiplication (KV) $\mathcal{O}(N^2 d)$

We can reorder calculations for efficient calculation.

$$\text{T-SM}(QK^\top)V = \ell_1 - \text{norm}\left[\sum_{n=0}^{2} \frac{(QK^\top)^n}{n!}\right] V = \text{norm}\left[\sum_{n=0}^{2} \frac{(QK^\top)^n}{n!} V \circ 1\right]$$

$$= \text{norm}\left[\frac{1}{2}(QK^\top)^2 (V \circ 1) + QK^\top (V \circ 1) + \sum_{\text{cols}} (V \circ 1)\right]$$

$$= \text{norm}\left[\frac{1}{2} Q^{\boxtimes 2} (K^{\boxtimes 2})^\top (V \circ 1) + QK(V \circ 1) + \sum_{\text{cols}} (V \circ 1)\right]$$

Efficient-TaylorShift

1. Matrix multiplication (QK)
2. Activation function
3. Normalization
4. Matrix multiplication (KV) $\mathcal{O}(N^2 d)$

1. Activation function
2. Matrix multiplication (KV)
3. Matrix multiplication (QK)
4. Normalization $\mathcal{O}(N d^3)$

We counteract numerical instabilities by introducing multiple normalizations.

Normalization at the end

⇒ large intermediate results

⇒ numerical instability

Expr.	$(K^{\boxtimes 2})^\top V = A_{\text{mod}}$	$(QK^T)^2 V$	$QK^\top V$	Y_{denom}	Y
Size	$\frac{N+1}{\sqrt{d}}$	$\frac{N}{d}$	$\sqrt{N}\frac{4d+1}{4d}$	$N\frac{d+2}{2d}$	$\sqrt{\frac{d}{N}}$

We counteract numerical instabilities by introducing multiple normalizations.

Normalization at the end

⇒ large intermediate results

⇒ numerical instability

1. Normalize queries and keys, use attention temperature

Expr.	$(K^{\boxtimes 2})^\top V = A_{\text{mod}}$	$(QK^T)^2 V$	$QK^\top V$	Y_{denom}	Y
Size	$\frac{N+1}{\sqrt{d}}$	$\frac{N}{d}$	$\sqrt{N}\frac{4d+1}{4d}$	$N\frac{d+2}{2d}$	$\sqrt{\frac{d}{N}}$

We counteract numerical instabilities by introducing multiple normalizations.

Normalization at the end

⇒ large intermediate results

⇒ numerical instability

1. Normalize queries and keys, use attention temperature
2. $(V \circ 1) \leftarrow \frac{1}{N}(V \circ 1)$

Expr.	$(K^{\boxtimes 2})^\top V = A_{\text{mod}}$	$(QK^T)^2 V$	$QK^\top V$	Y_{denom}	Y
Size	$\frac{N+1}{N}\frac{1}{\sqrt{d}}$	$\frac{1}{d}$	$\frac{1}{\sqrt{N}}\frac{4d+1}{4d}$	$\frac{d+2}{2d}$	$\sqrt{\frac{d}{N}}$

We counteract numerical instabilities by introducing multiple normalizations.

Normalization at the end

⇒ large intermediate results

⇒ numerical instability

1. Normalize queries and keys, use attention temperature
2. $(V \circ 1) \leftarrow \frac{1}{N}(V \circ 1)$
3. $Q, K \leftarrow \sqrt[4]{d}Q, \sqrt[4]{d}K$, adjust factors accordingly

Expr.	$(K^{\boxtimes 2})^\top V = A_{\mathrm{mod}}$	$(QK^T)^2 V$	$QK^\top V$	Y_{denom}	Y
Size	$\frac{N+1}{N}$	1	$\frac{\sqrt{d}}{\sqrt{N}} + \frac{1}{4\sqrt{Nd}}$	$\frac{d+2}{2d}$	$\sqrt{\frac{d}{N}}$

We counteract numerical instabilities by introducing multiple normalizations.

Normalization at the end

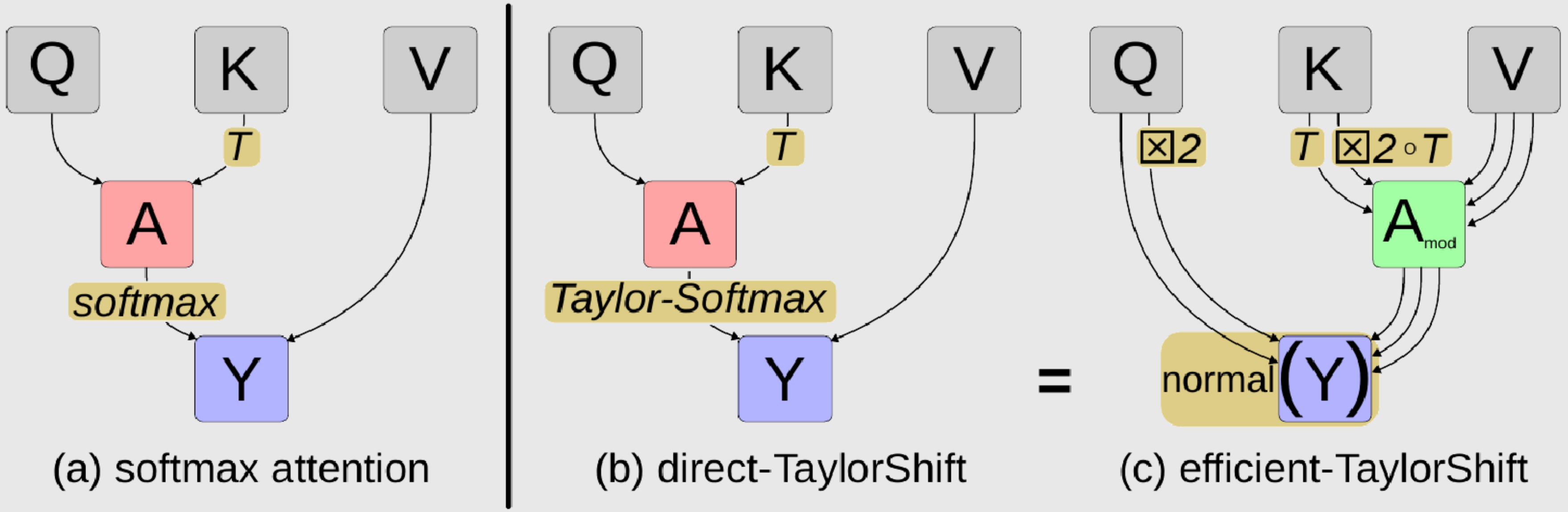
⇒ large intermediate results

⇒ numerical instability

1. Normalize queries and keys, use attention temperature
2. $(V \circ 1) \leftarrow \frac{1}{N}(V \circ 1)$
3. $Q, K \leftarrow \sqrt[4]{d}Q, \sqrt[4]{d}K$, adjust factors accordingly
4. $Y \leftarrow \sqrt{\frac{N}{d}}Y$

Expr.	$(K^{\boxtimes 2})^\top V = A_{\text{mod}}$	$(QK^T)^2V$	$QK^\top V$	Y_{denom}	Y
Size	$\frac{N+1}{N}$	1	$\frac{\sqrt{d}}{\sqrt{N}} + \frac{1}{4\sqrt{Nd}}$	$\frac{d+2}{2d}$	1

TaylorShift calculates token-to-token interactions in linear time.



1. Softmax → Taylor-Softmax
2. Reorder operations for efficient calculation
3. Normalize intermediate results

TaylorShift is efficient for long sequences, depending on d.

When is $O(N d^3)$ actually better than $O(N^2 d)$?

	Speed
Proxy Metric	# FLOPs

TaylorShift is efficient for long sequences, depending on d.

When is $O(N d^3)$ actually better than $O(N^2 d)$?

	Speed
Proxy Metric	# FLOPs
Direct-TaylorShift	$\mathrm{ops}_{\mathrm{dir}} = 4N^2 d + 6N^2$
Efficient-TaylorShift	$\mathrm{ops}_{\mathrm{eff}} = N(4d^3 + 10d^2 + 9d + 4)$

TaylorShift is efficient for long sequences, depending on d.

When is O(N d^3) actually better than O(N^2 d)?

	Speed
Proxy Metric	# FLOPs
Direct-TaylorShift	$\mathrm{ops}_{\mathrm{dir}} = 4N^2 d + 6N^2$
Efficient-TaylorShift	$\mathrm{ops}_{\mathrm{eff}} = N(4d^3 + 10d^2 + 9d + 4)$
Intersection length	$N_0 \leq d^2 + d + 1$

TaylorShift is efficient for long sequences, depending on d.

When is $O(N\,d^3)$ actually better than $O(N^2\,d)$?

	Speed	**Memory**
Proxy Metric	# FLOPs	# Entries in intermediate results
Direct-TaylorShift	$\text{ops}_{\text{dir}} = 4N^2d + 6N^2$	$\text{entr}_{\text{dir}} = Nd + 2N^2$
Efficient-TaylorShift	$\text{ops}_{\text{eff}} = N(4d^3 + 10d^2 + 9d + 4)$	$\text{entr}_{\text{eff}} = d^2(d+1) + 2Nd + N(d+1) + Nd^2$
Intersection length	$N_0 \leq d^2 + d + 1$	

TaylorShift is efficient for long sequences, depending on d.

When is $O(N\,d^3)$ actually better than $O(N^2\,d)$?

	Speed	**Memory**
Proxy Metric	# FLOPs	# Entries in intermediate results
Direct-TaylorShift	$\mathrm{ops}_{\mathrm{dir}} = 4N^2d + 6N^2$	$\mathrm{entr}_{\mathrm{dir}} = Nd + 2N^2$
Efficient-TaylorShift	$\mathrm{ops}_{\mathrm{eff}} = N(4d^3 + 10d^2 + 9d + 4)$	$\mathrm{entr}_{\mathrm{eff}} = d^2(d+1) + 2Nd + N(d+1) + Nd^2$
Intersection length	$N_0 \leq d^2 + d + 1$	$N_1 \leq \frac{1}{2}d^2 + 2d + \frac{1}{2}$

TaylorShift is efficient for long sequences, depending on d.

When is O(N d^3) actually better than O(N^2 d)?

	Speed	**Memory**
Proxy Metric	# FLOPs	# Entries in intermediate results
Direct-TaylorShift	$\mathrm{ops}_{\mathrm{dir}} = 4N^2 d + 6N^2$	$\mathrm{entr}_{\mathrm{dir}} = Nd + 2N^2$
Efficient-TaylorShift	$\mathrm{ops}_{\mathrm{eff}} = N(4d^3 + 10d^2 + 9d + 4)$	$\mathrm{entr}_{\mathrm{eff}} = d^2(d+1) + 2Nd + N(d+1) + Nd^2$
Intersection length	$N_0 \le d^2 + d + 1$	$N_1 \le \frac{1}{2} d^2 + 2d + \frac{1}{2}$

d	8	16	32	64	128
N_0	73	273	1057	4161	16513
N_1	47	159	574	2174	8446

Increasing the number of attention-heads increases TaylorShift's efficiency.

Reducing the dimension d makes TaylorShift more efficient.

Reduces the representational capacity of the model.

Increasing the number of attention-heads increases TaylorShift's efficiency.

Reducing the dimension d makes TaylorShift more efficient.

Reduces the representational capacity of the model.

Change the number of attention-heads instead.

$$d = \frac{d_{\text{embed}}}{h} \qquad h \times \text{ops}_{\text{eff}} \qquad h \times \text{entr}_{\text{eff}}$$

$$h \nearrow \qquad \searrow \qquad \searrow$$

Increasing the number of attention-heads increases TaylorShift's efficiency.

Reducing the dimension d makes TaylorShift more efficient.

Reduces the representational capacity of the model.

Change the number of attention-heads instead.

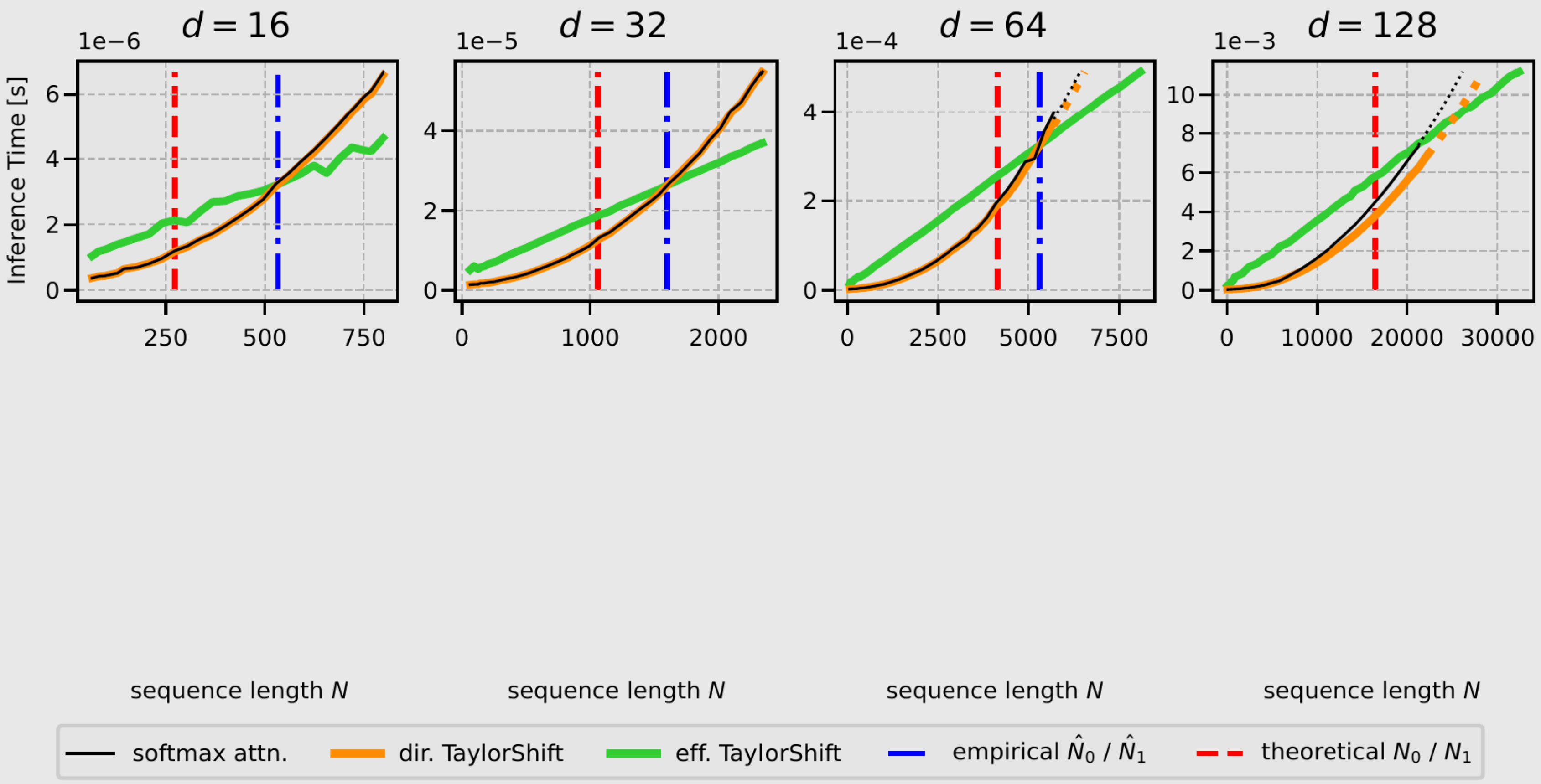
$$d = \frac{d_{\text{embed}}}{h} \qquad h \times \text{ops}_{\text{eff}} \qquad h \times \text{entr}_{\text{eff}}$$

$$h \nearrow \qquad \searrow \qquad \searrow$$

⇒ Increase efficiency without decreasing overall capacity by increasing the number of heads

TaylorShift is efficient for long sequences, as predicted.

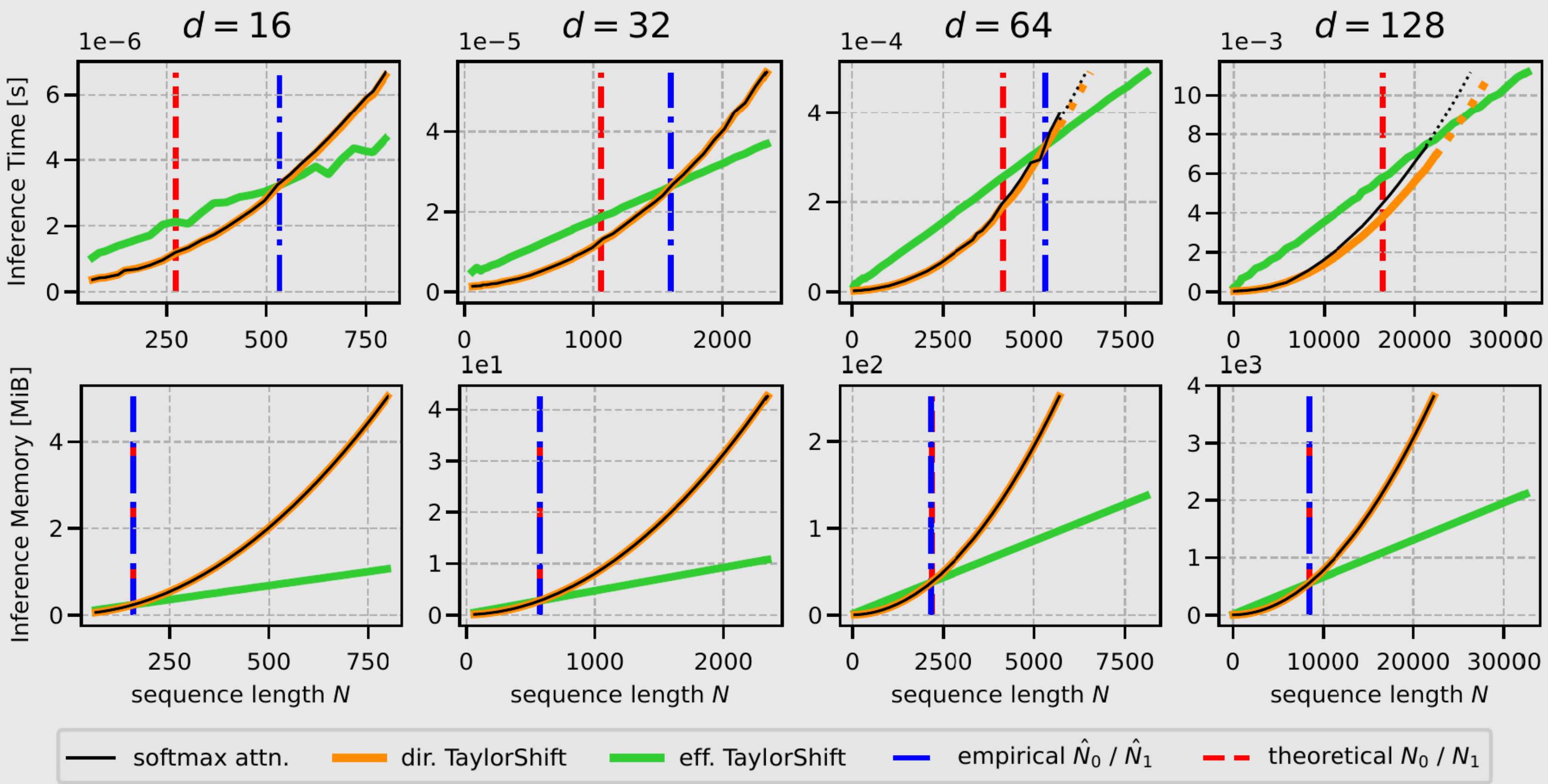
Speed:
- Direct-TaylorShift outperforms attention
- Intersection point is larger than predicted
- Slow memory reads on current hardware



TaylorShift is efficient for long sequences, as predicted.

Speed:
- Direct-TaylorShift outperforms attention
- Intersection point is larger than predicted
- Slow memory reads on current hardware

Memory:
- Almost exactly like predicted theoretically



TaylorShift outperforms standard attention in 4/5 tasks.

TaylorShift outperforms the other models, including the baseline Transformer encoder, in (at least) 4/5 tasks.

Model	CIFAR (Pixel)	IMDB (Byte)	ListOps	ImageNet (Ti)	ImageNet (S)	Average
Linformer	29.2	58.1	-	64.3	76.3	(57.0)
RFA	44.9	65.8	-	-	-	(55.4)
Performer	34.2*	65.6*	35.4*	62.0*	67.1*	52.9
Reformer	44.8	63.9	**47.6**	73.6	76.2*	61.2
Nystromformer	**49.4**	65.6	44.5	75.0	78.3*	62.6
EVA	46.1	64.0	45.3	73.4	78.2	61.4
Transformer	44.7	65.8	46.0	**75.6**	79.1	62.2
Ours	47.6	**66.2**	46.1	75.0	**79.3**	**62.8**

Increasing the number of heads speeds up TaylorShift and can increase accuracy.

Increasing the number of heads:
- Reduces memory consumption
- Increases speed
- Sometimes increases accuracy

h	d	Acc [%]	direct TP [ims/s]	direct Mem [MiB@16]	efficient TP [ims/s]	efficient Mem [MiB@16]
4	64	47.1	12 060	596	2 975	840
8	32	47.5	7 657	1 111	5 749	585
16	16	47.3	4 341	2 135	9 713	459
32	8	46.9	2 528	4 187	14 087	397
64	4	45.9	1 235	8 291	13 480	125

Accuracy, throughput and VRAM usage of TaylorShift on the CIFAR Pixel task with increased number of attention heads with:

$d_{\text{embed}} = d \times h = 256$ $\qquad N = 1024$

TaylorShift:
Shifting the Complexity of Self-Attention from Squared to Linear (and Back) using Taylor-Softmax

Website

PDF

Code