Efficient LLM Pretraining & Inference with Unlimited Context Length

Xuezhe Ma (Max)

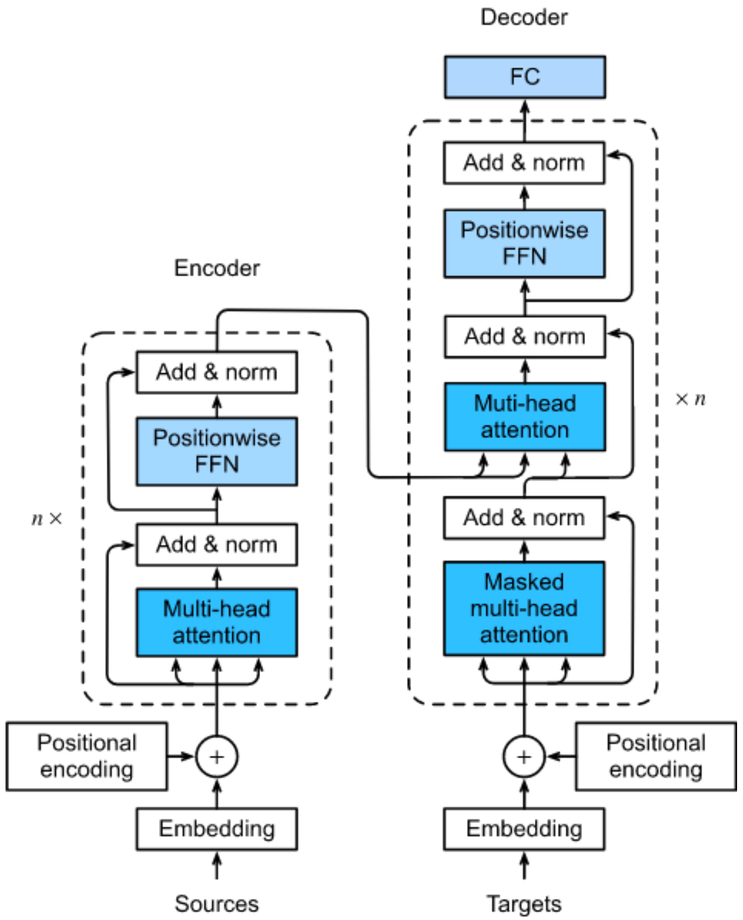
collaborators at

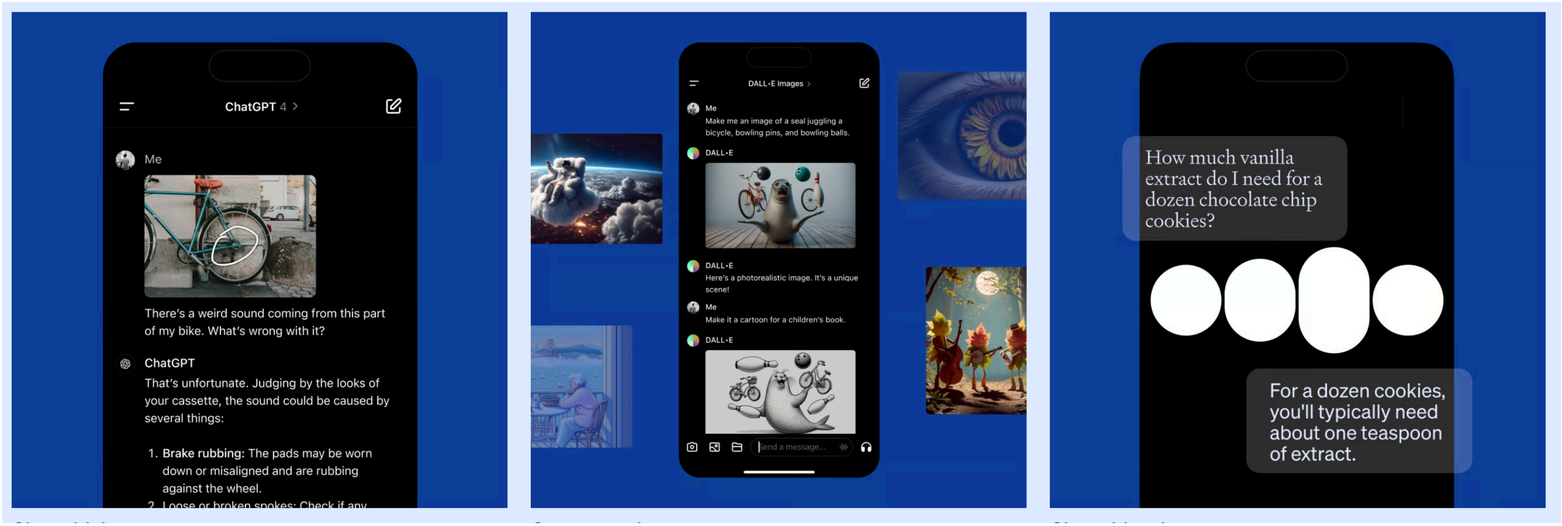
1. USC-ISI 2. Meta AI 3. CMU 4. UCSD 5. MIT 6. SJTU

Carnegie Mellon University
Language Technologies Institute

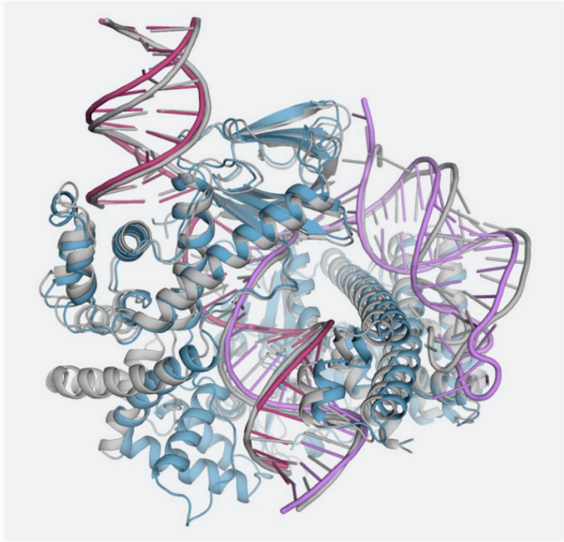
上海交通大學
Shanghai Jiao Tong University

Transformer is the Backbone of Most ML Models

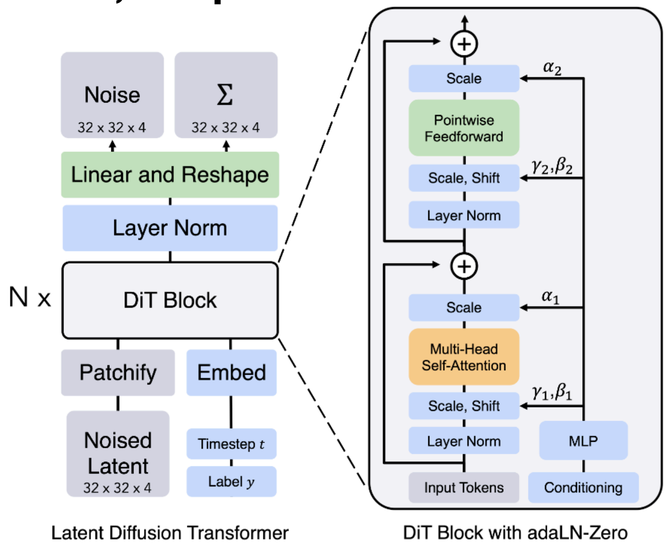




ChatGPT, GPT4V, Wisper



AlphaFold



Latent Diffusion Transformer

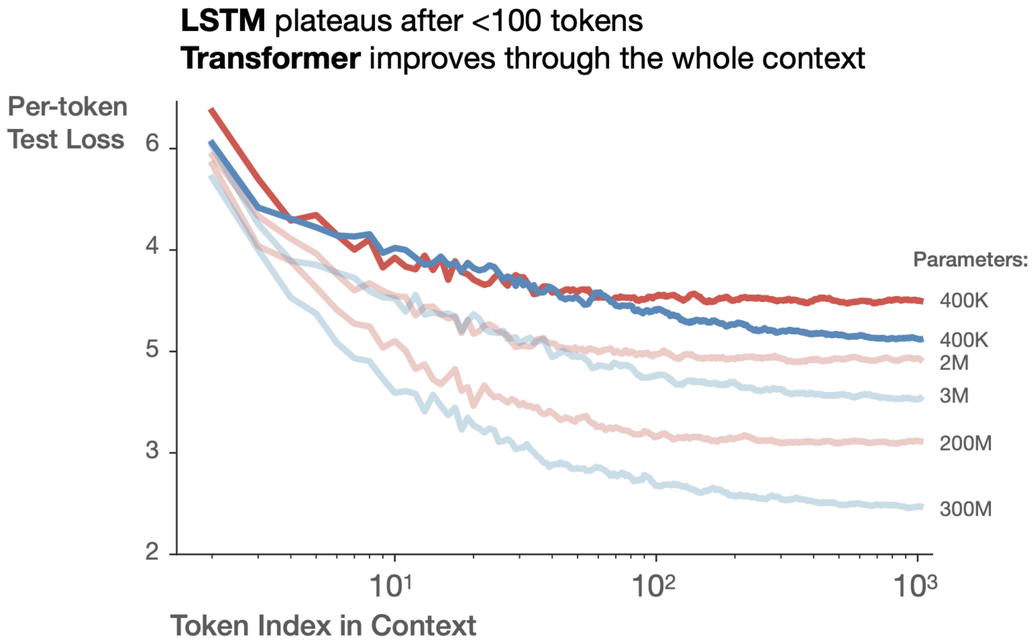
DiT Block with adaLN-Zero

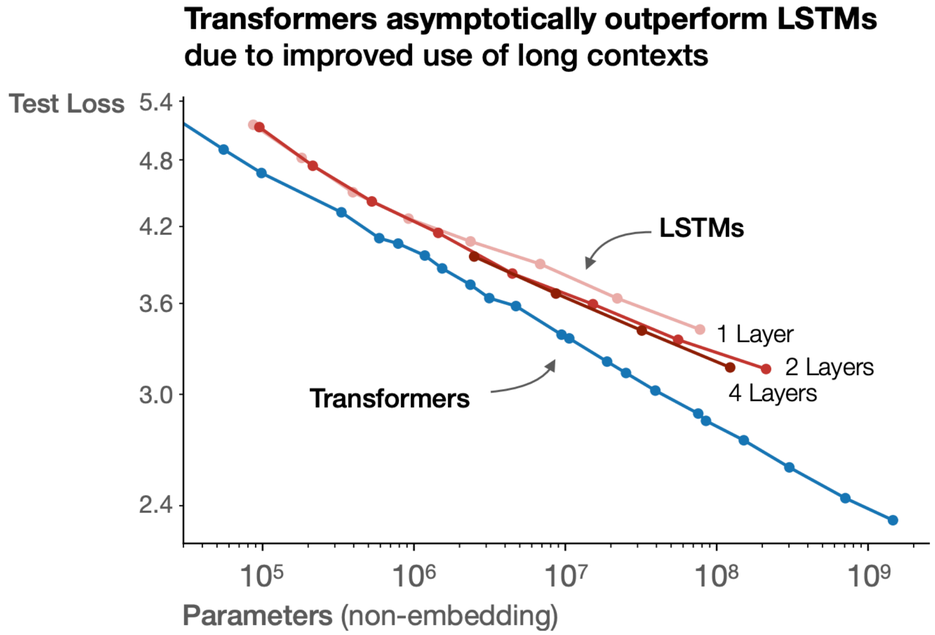
Transformer Diffusion Model (DiT)

Why Long Context?

- Longer Context = Higher Intelligence + Better Performance

Transformers Scale Better than LSTMs due to Improved Use of Context





“Better use of context” → **“Better model performance”**

Scaling laws for neural language models. Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... & Amodei, D. (2020)

Slide modified from Hao Liu’s talk

Why Long Context?

- Longer Context = Higher Intelligence + Better Performance
- Long context unlocks more applications

book-level translation/ summarization

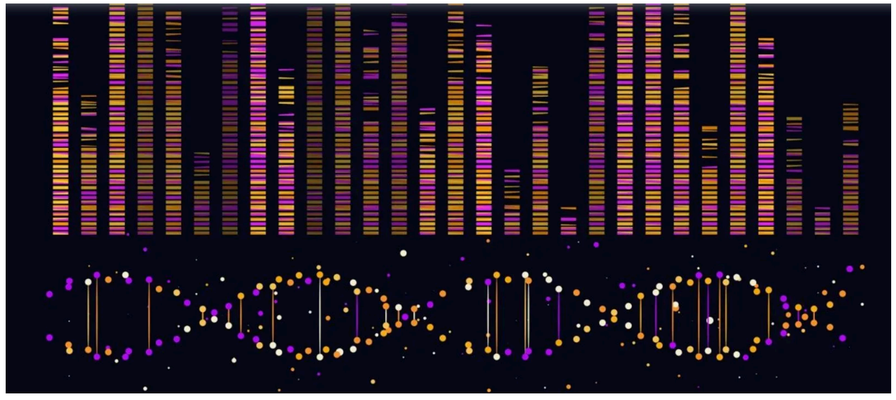
personalized AI assistant

videos and world models

```
"SPLN_Code_Comment":{
    "prefix": "SPLN_Code_Comment",
    "body": ["$LINE_COMMENT SPLN${1:1}.00 - Sta
             "\t$TM_SELECTED_TEXT",
             "$LINE_COMMENT SPLN${1:1}.00 - END"
    ],
    "description": "Comment for Code changes"
},

"SPLN_Documentation_Comment":{
    "prefix": "SPLN_Documentation_Comment",
    "body": ["\t$LINE_COMMENT SPLN${1:1}.00 ${2
             "\t$LINE_COMMENT\t\t\t${4:Code modi
             "\t$LINE_COMMENT\t\t\t - ${5:Functi
    ],
    "description": "Comment for Code changes (d
}
```

code and hyperlinks

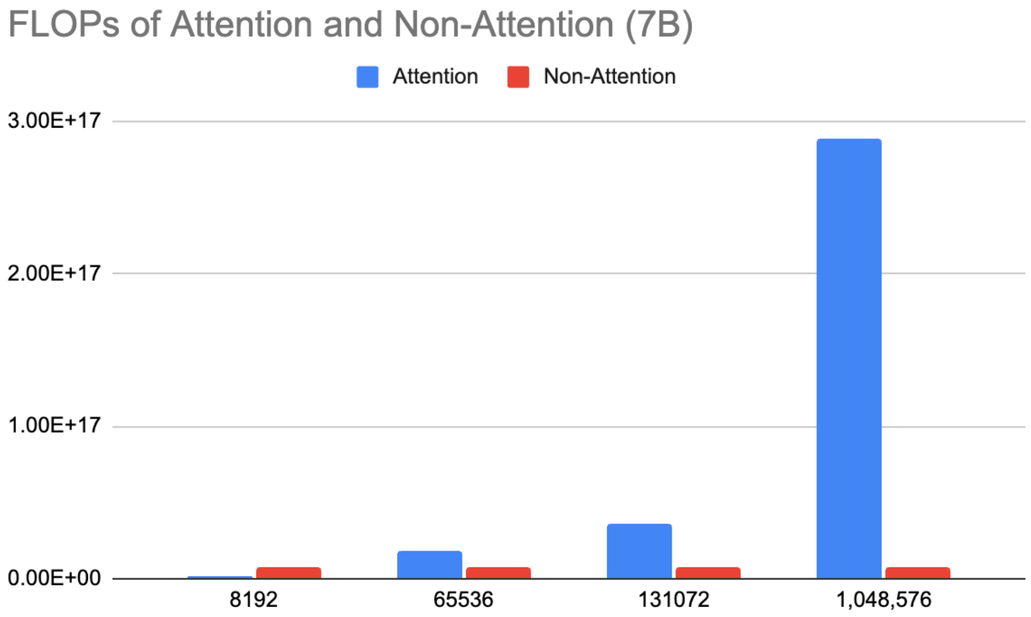


Genome

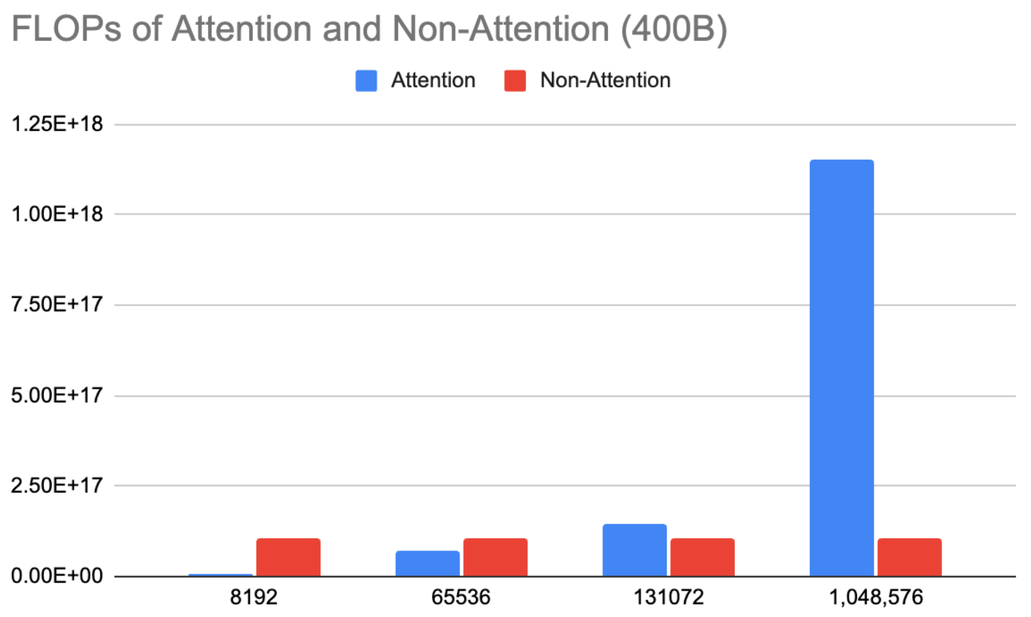
Long Context is hard for Transformer

- **Complexity**
  - Expensive: quadratic complexity
    - Both time and space
    - $O(hn^2)$: h heads and sequence length of n

FLOPs of Attention and Non-attention Operations



sequence length

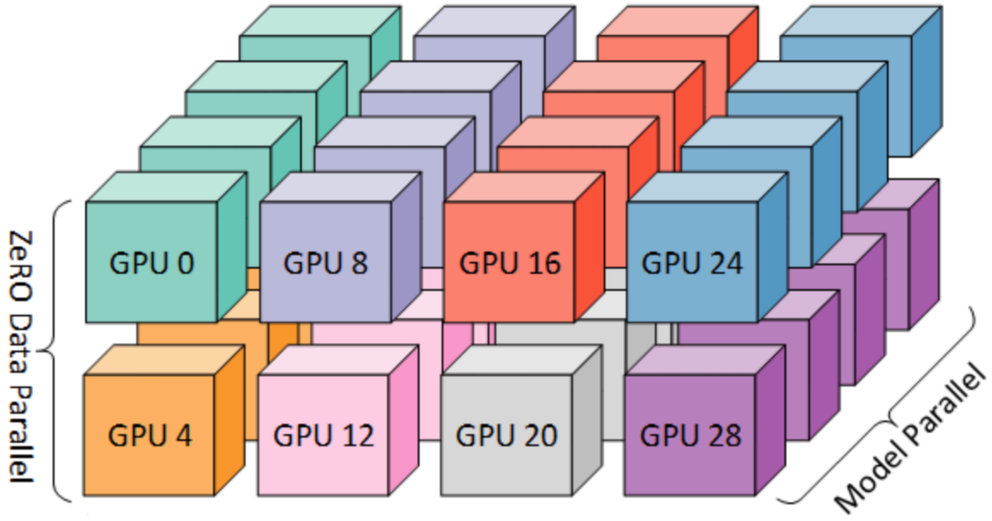


sequence length

FLOPs of attention is not negligible even for very large models!

Full Attention: quadratic Complexity

- **It is hard to do distributed training across the sequence dimension on multiple devices with Transformers**



Even with efficient attention or advanced sequence parallel implementations:

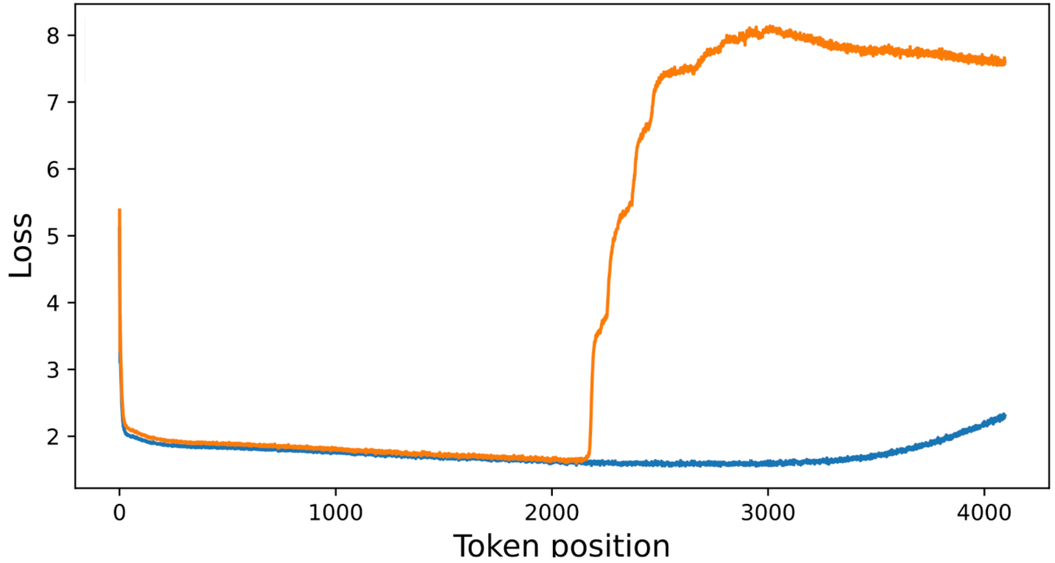
- Memory cost (*2bnh*) scales linearly with sequence length *n* (e.g. flash attention)
- Huge communication overhead and FLOPs (e.g. ring attention, context parallelism, etc)
- KV cache takes huge memory at inference time when the prompt is very long

Image modified from Microsoft blog

Long Context is hard for Transformer

- **Complexity**
  - Expensive: quadratic complexity
    - Both time and space
    - $O(hn^2)$: h heads and sequence length of n

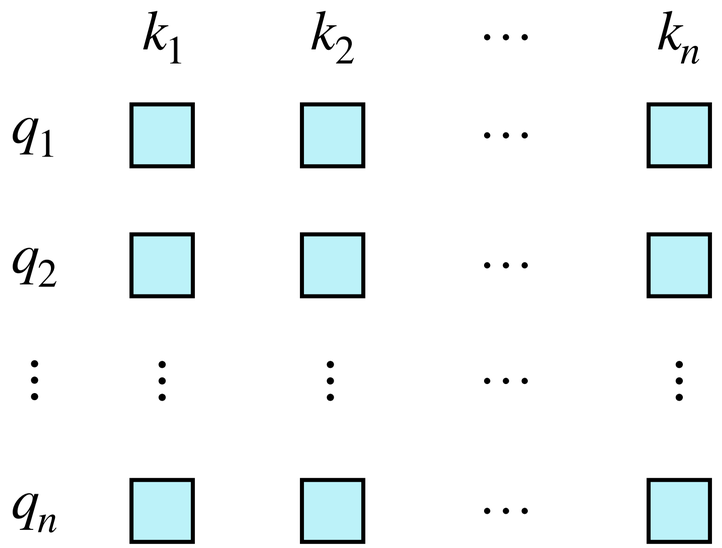
- **Inductive Bias on Length Generalization**
  - Hard to be extended to longer contexts than pre-training



Weak Inductive Bias on Length Generalization

- **Transformer Hard to be extended to longer contexts than pre-training**

Attention score $A \in \mathbb{R}^{n \times n}$

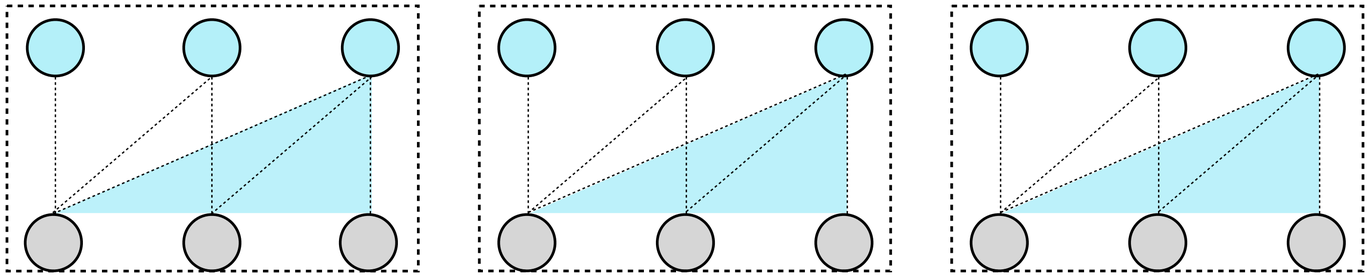


- Pair-wise interaction
- Order-invariant if no positional embeddings

Why not Chunk-wise Attention?

- **Applying attention individually to each chunk with fixed length**

chunk-wise Attention



Pros

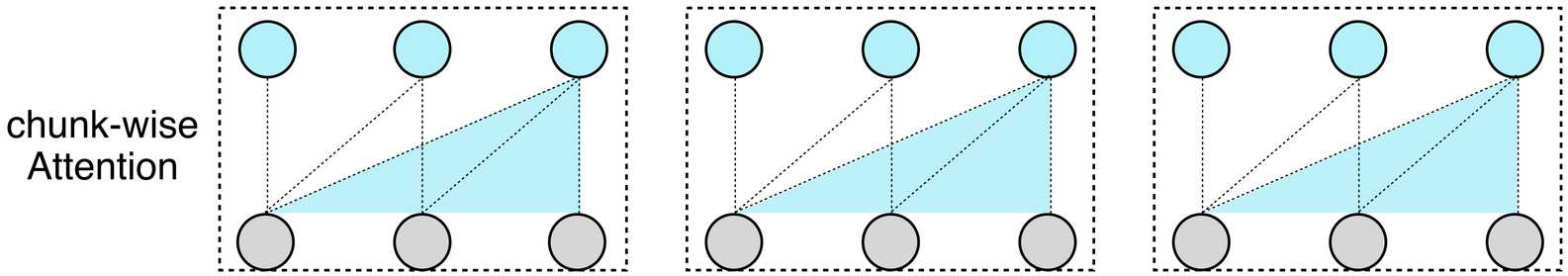
- Efficiency: Linear complexity
- Inductive bias: No need for length extension

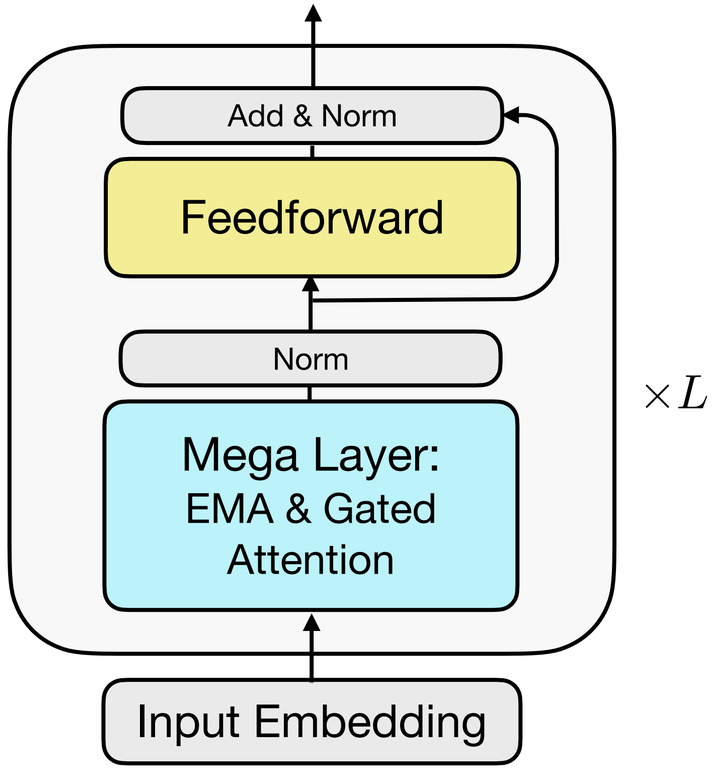
Cons

- Capacity: Limited contexts

Mega: Executive Summary

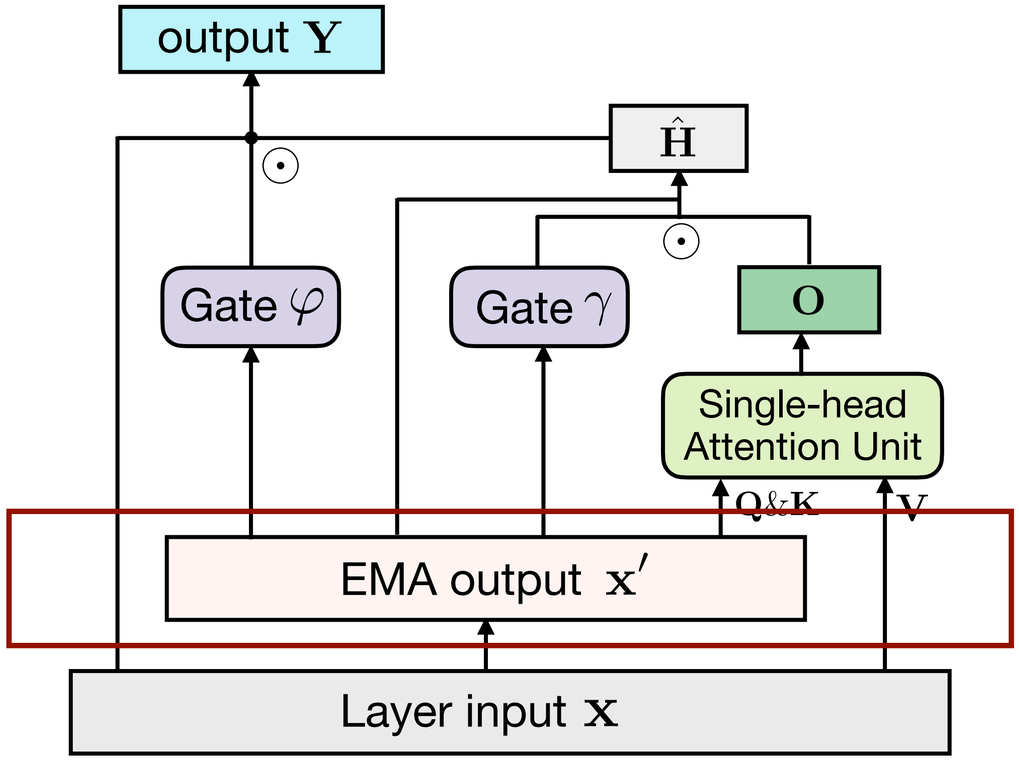
- **Effective and efficient drop-in replacement of attention for long sequence modeling**
  - Outstanding results on various types of data
    - text, images and audios
  - Exponential Moving Average (EMA)
  - Mega-chunk: linear complexity of time and space





Mega Architecture: Outline

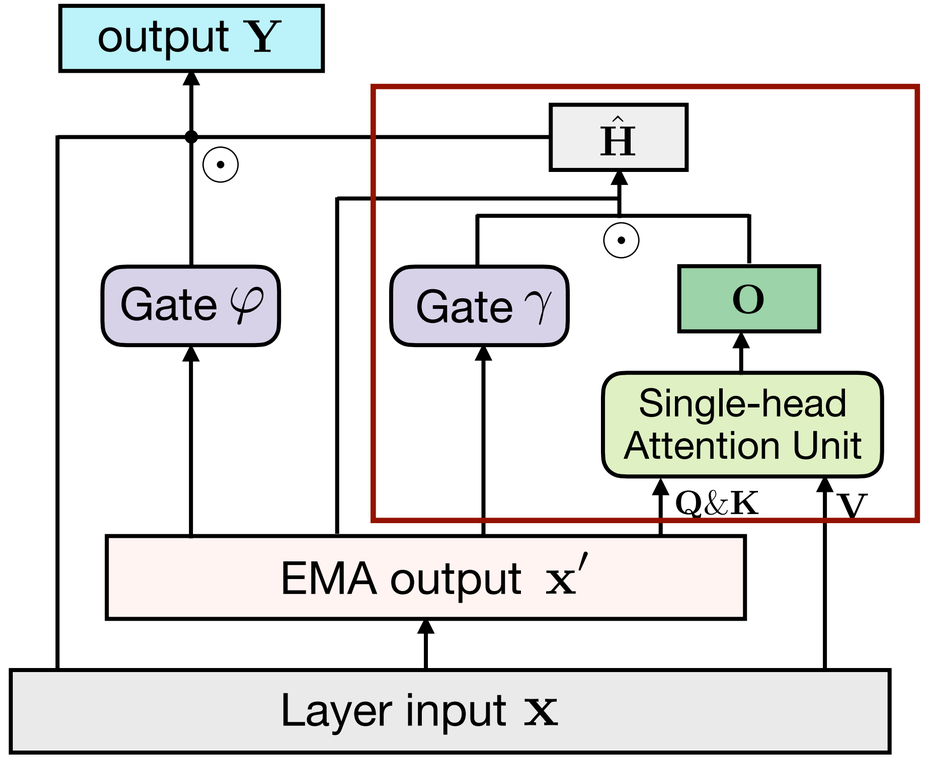
- **Exponential Moving Average (EMA)**
  - Local dependencies that decaying exponentially over time



Mega Architecture: Outline

- **Exponential Moving Average (EMA)**
  - Local dependencies that decaying exponentially over time

- **Single-head Gated Attention**
  - Adding a reset gate to the attention output
  - Theoretically proving that single-head gated attention is as expressive as multi-head one

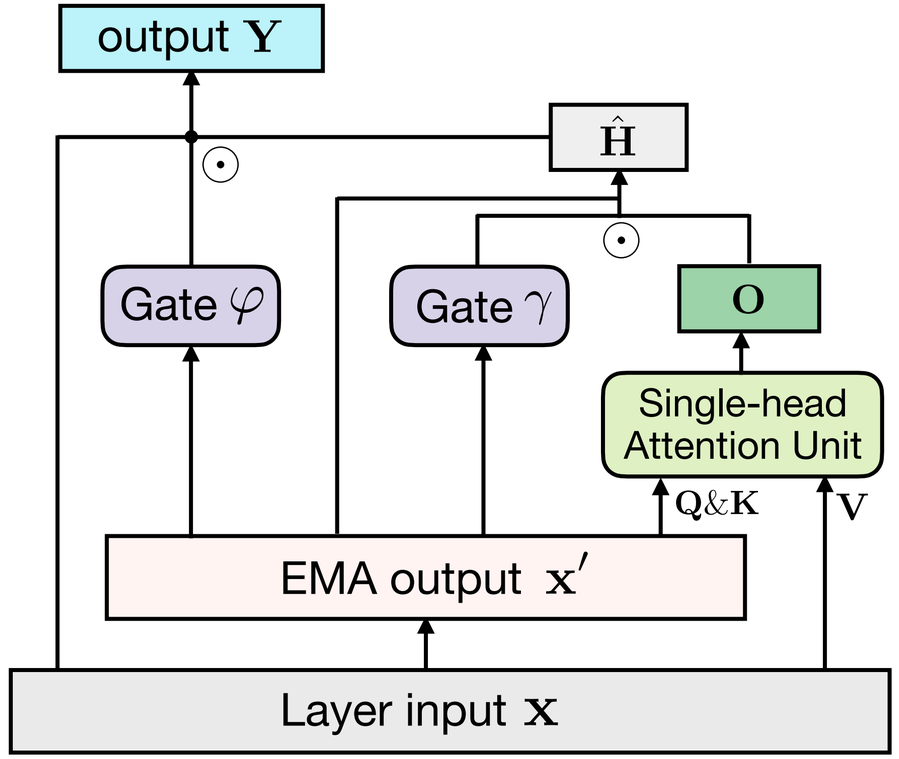


Mega Architecture: Outline

- **Exponential Moving Average (EMA)**
  - Local dependencies that decaying exponentially over time

- **Single-head Gated Attention**
  - Adding a reset gate to the attention output
  - Theoretically proving that single-head gated attention is as expressive as multi-head one

- **Mega-Chunk**
  - Applying attention to local chunks of fixed length
  - Reducing quadratic complexity to linear

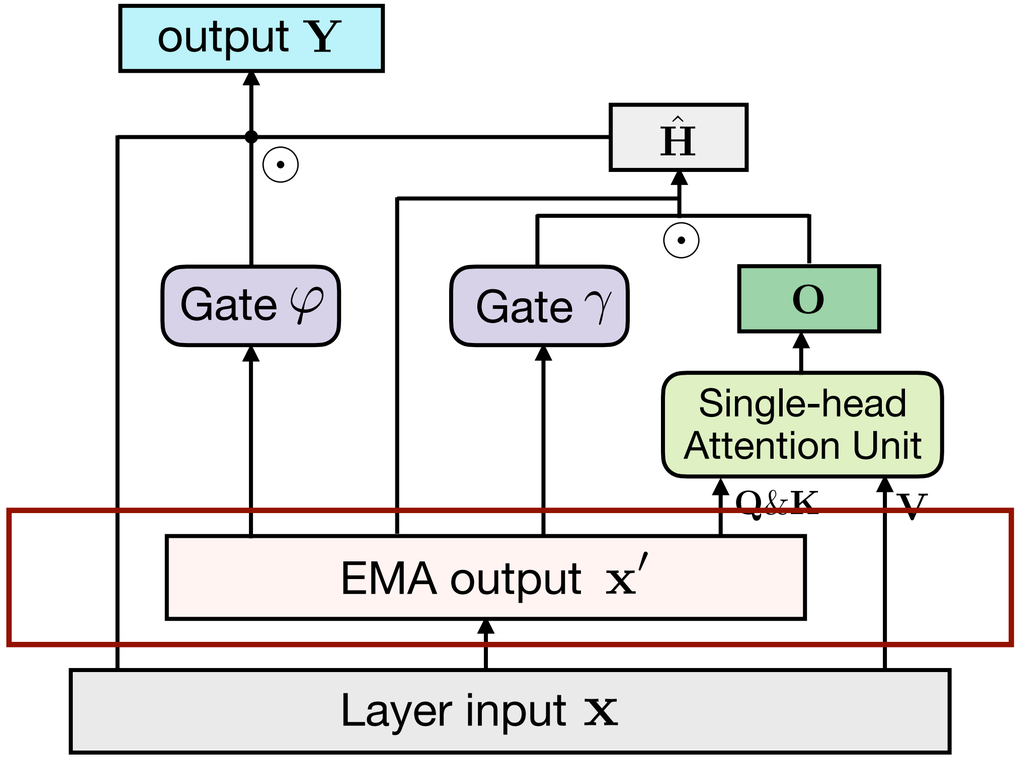


Mega Architecture: Outline

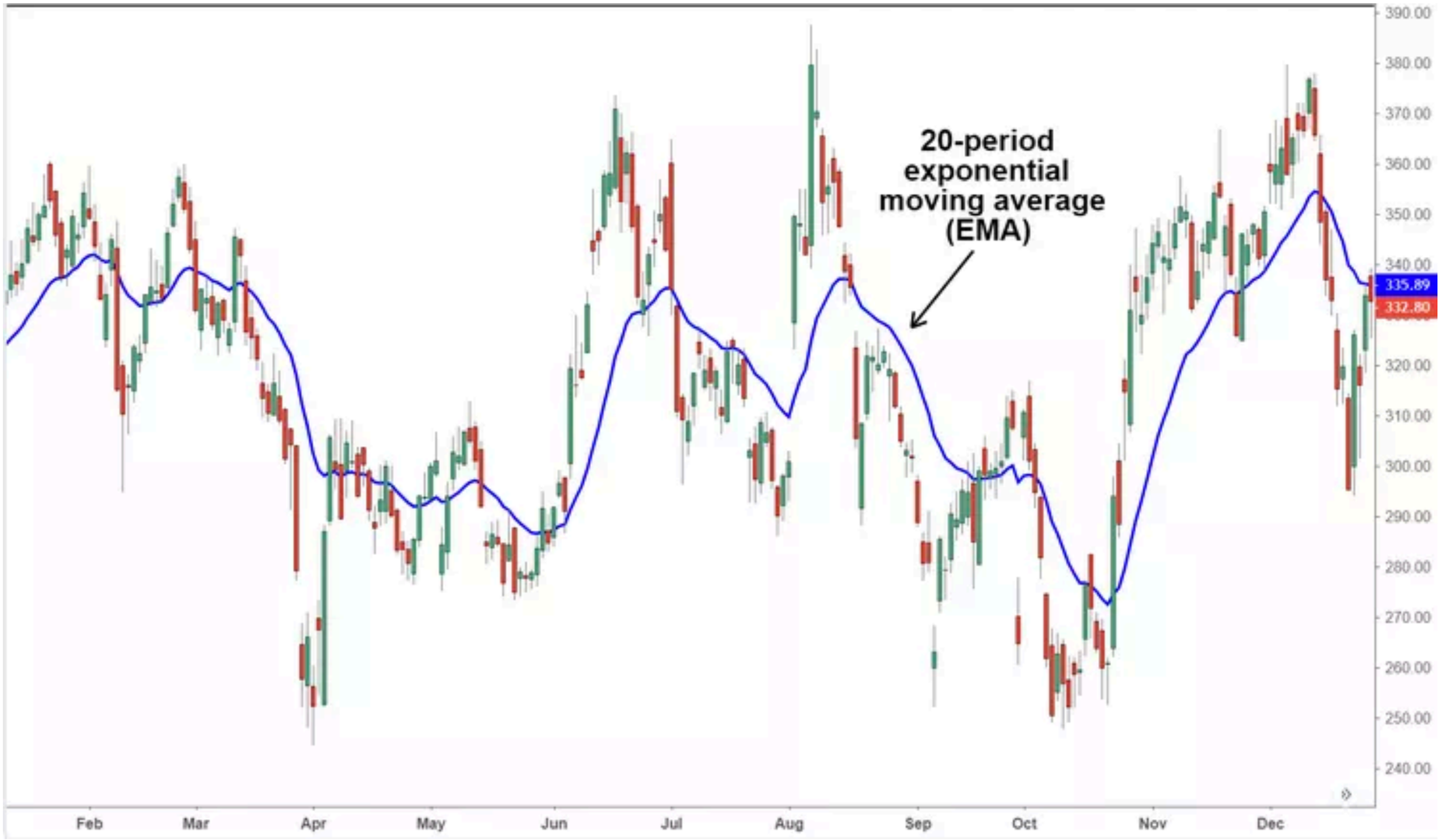
- **Exponential Moving Average (EMA)**
  - Local dependencies that decaying exponentially over time

- **Single-head Gated Attention**
  - Adding a reset gate to the attention output
  - Theoretically proving that single-head gated attention is as expressive as multi-head one

- **Mega-Chunk**
  - Applying attention to local chunks of fixed length
  - Reducing quadratic complexity to linear



Exponential Moving Average (EMA)



Exponential Moving Average (EMA)

Notations: Assuming 1-dim input sequence $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_t \in \mathbb{R}$

EMA

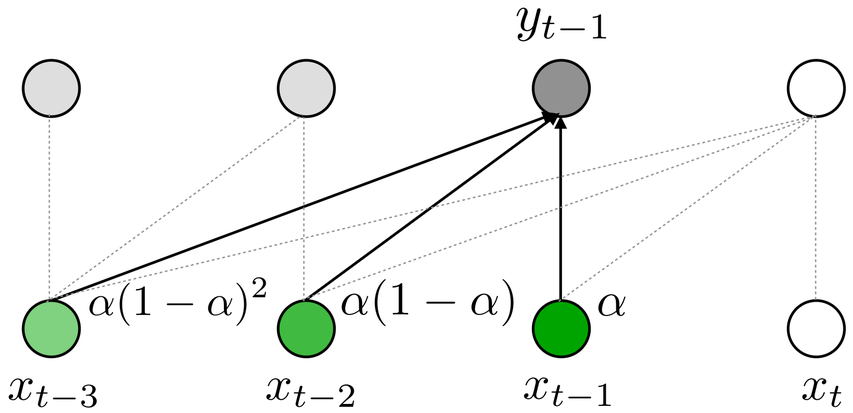
$$\mathbf{y}_t = \boldsymbol{\alpha} \odot \mathbf{x}_t + (1 - \boldsymbol{\alpha}) \odot \mathbf{y}_{t-1}, \quad \boldsymbol{\alpha} \in (0, 1)$$

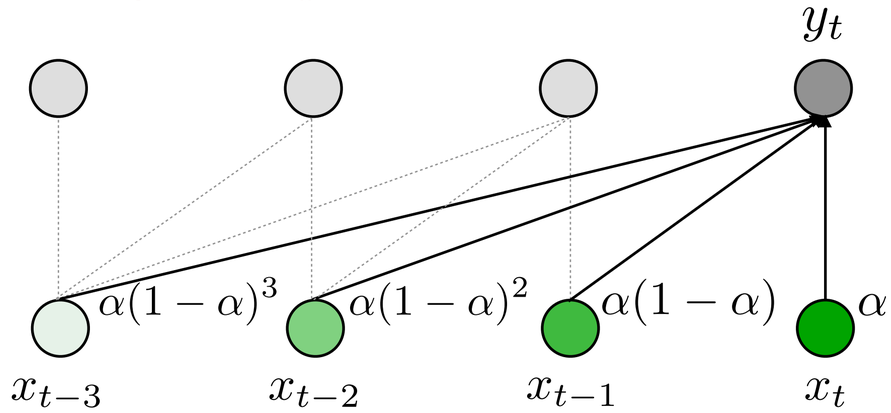
Damped EMA

$$\mathbf{y}_t = \boldsymbol{\alpha} \odot \mathbf{x}_t + (1 - \boldsymbol{\alpha} \odot \boldsymbol{\delta}) \odot \mathbf{y}_{t-1}, \quad \boldsymbol{\delta} \in (0,1)$$

Learnable Parameters

Relaxing the coupled weights



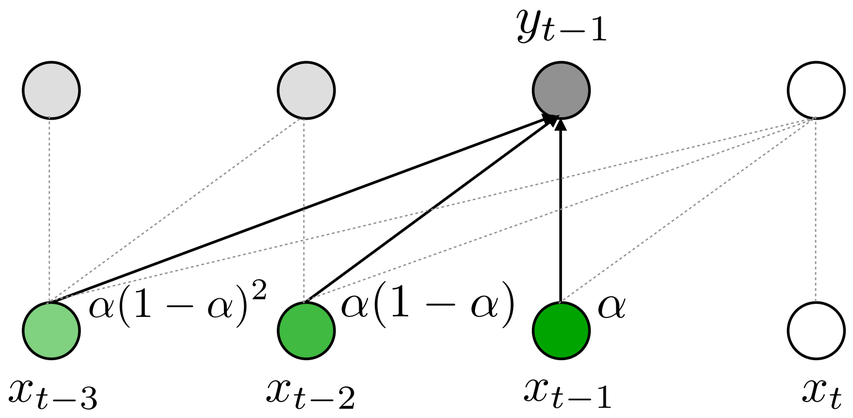


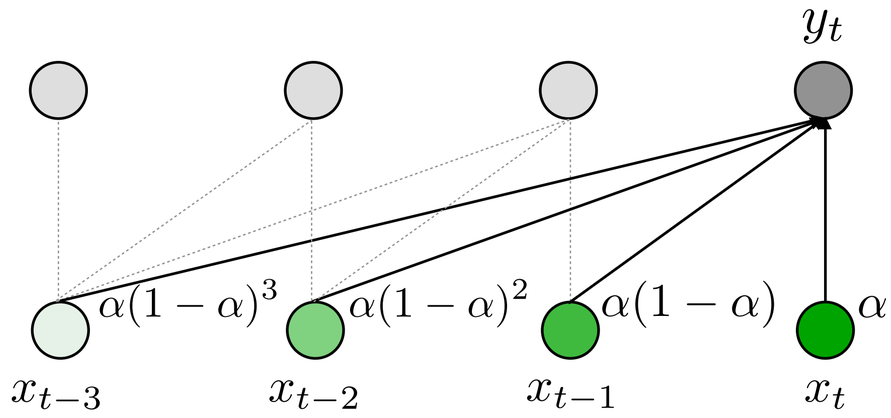
Efficient Algorithm for EMA

- **Efficiently compute EMA outputs of all tokens in parallel**

$$\mathbf{y}_t = \boxed{\boldsymbol{\alpha}} \odot \mathbf{x}_t + \boxed{(1 - \boldsymbol{\alpha} \odot \boldsymbol{\delta})} \odot \mathbf{y}_{t-1}$$

EMA weights are input independent

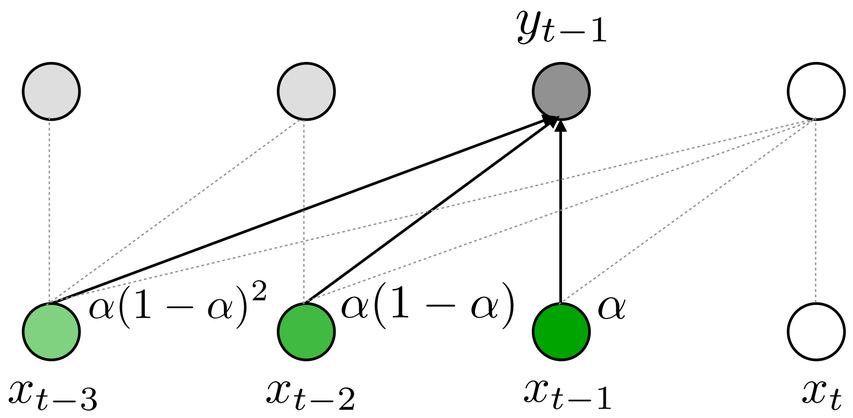


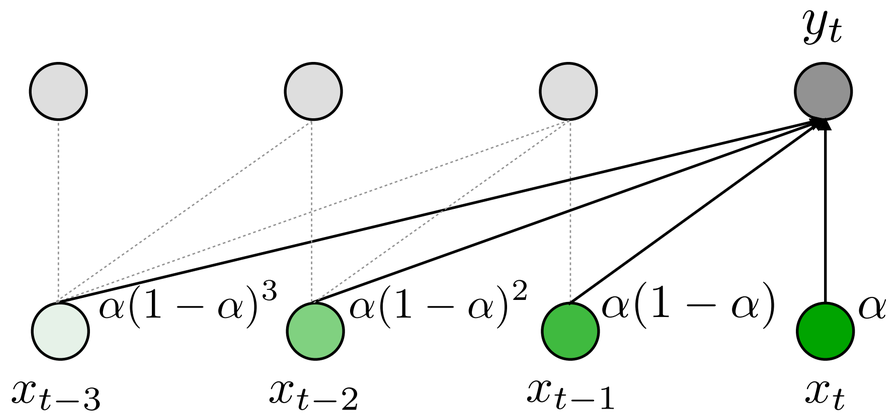


We can compute the weights for each input tokens in advance and compute EMA with FFTs.

EMA: Summary

- **Strong Inductive Bias**
  - Favors recent, local contexts
  - In principle, can peek unbounded context history
- **Efficient to compute and easy to learn**
  - Parallel computation
  - Less parameters & capacity
  - Extremely simplified state space model



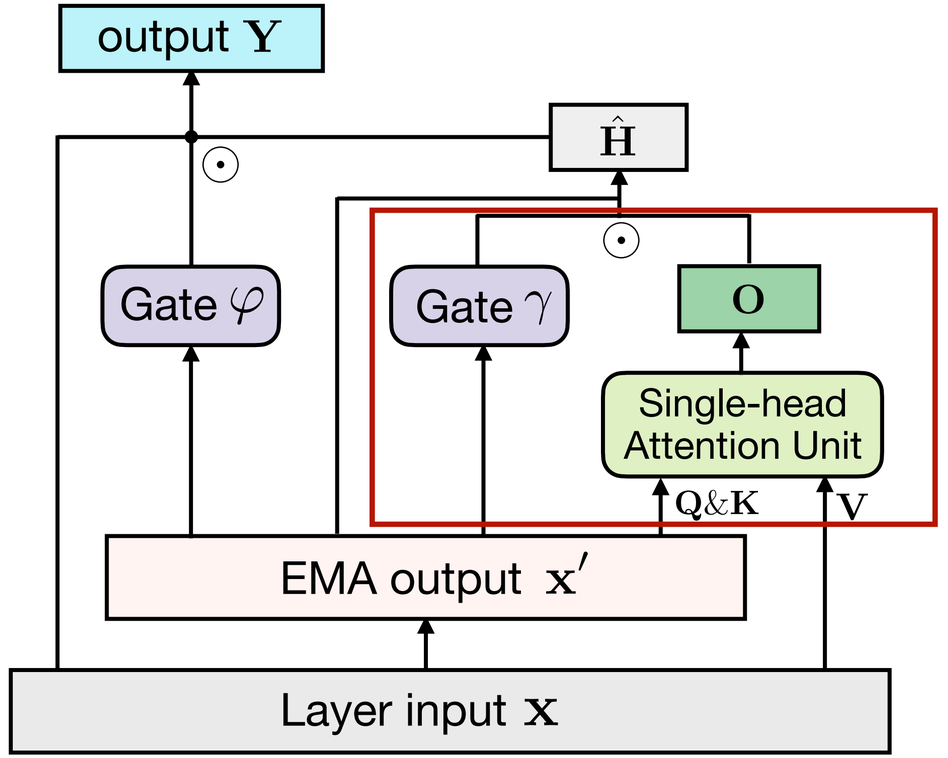


Mega Architecture: Outline

- Exponential Moving Average (EMA)
  - Local dependencies that decaying exponentially over time
  - Connection with State Space Models (e.g S4)

- **Single-head Gated Attention**
  - Adding a reset gate to the attention output
  - Theoretically proving that single-head gated attention is as expressive as multi-head one

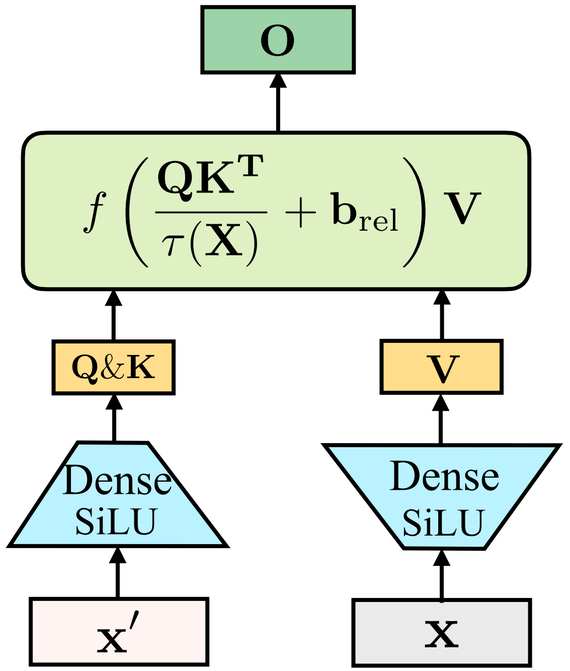
- Mega-Chunk
  - Applying attention to local chunks of fixed length
  - Reducing quadratic complexity to linear



Single-head Gated Attention

- **Adding a reset gate to the attention output**

Attention: $$O = \text{softmax}\left(\frac{QK^T}{\tau(X)} + b_{\text{rel}}\right)V$$



Single-head Gated Attention

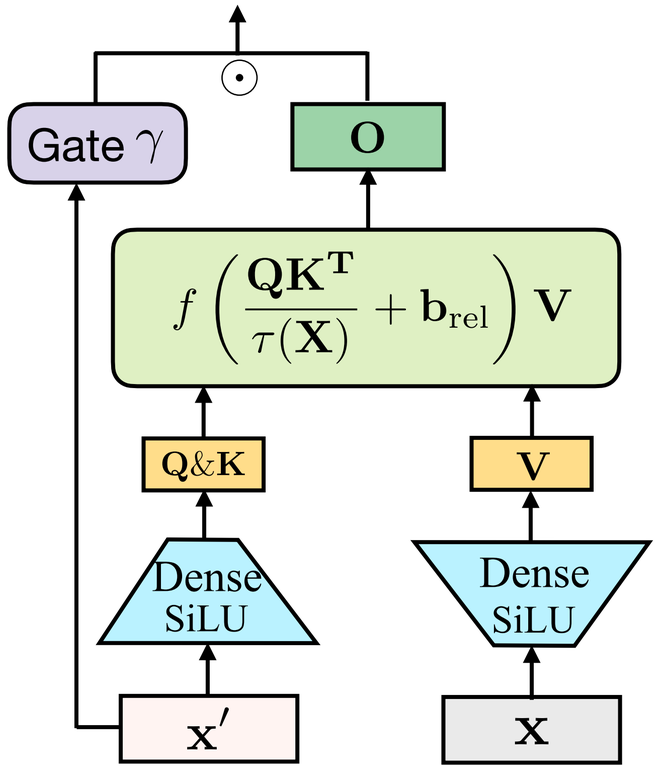
- **Adding a reset gate to the attention output**

Attention: $$O = \text{softmax}\left(\frac{QK^T}{\tau(X)} + b_{\text{rel}}\right)V$$

Gated Attention: $$\gamma = \mathcal{G}(X)$$

$$O_{\text{SHGA}} = O \odot \gamma$$

$\mathcal{G}()$ is a transform, e.g. an MLP



Single-head Gated Attention

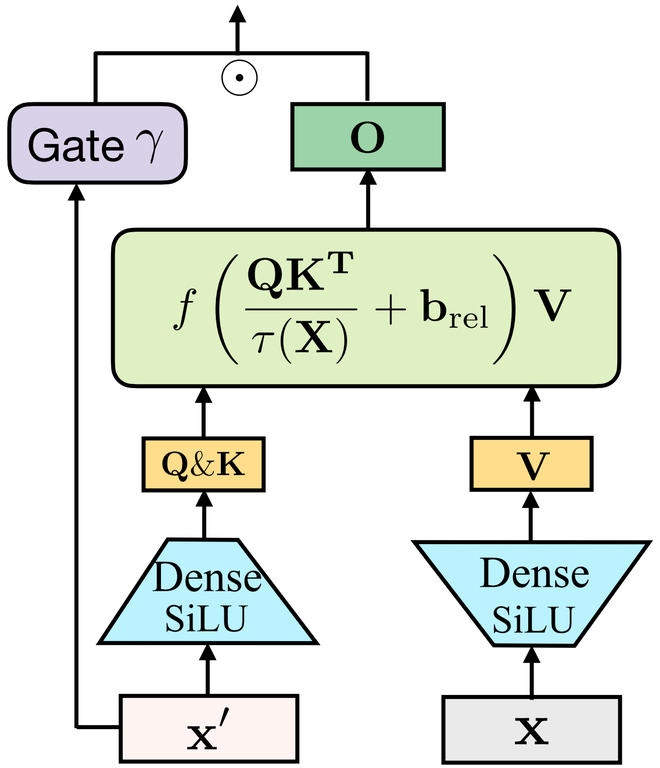
- **Adding a reset gate to the attention output**

Attention: $$O = \text{softmax}\left(\frac{QK^T}{\tau(X)} + b_{\text{rel}}\right)V$$

Gated Attention: $$\gamma = \mathcal{G}(X)$$

$$O_{\text{SHGA}} = O \odot \gamma$$

$\mathcal{G}()$ is a transform, e.g. an MLP



- **Single-head gated attention is as expressive as multi-head one**

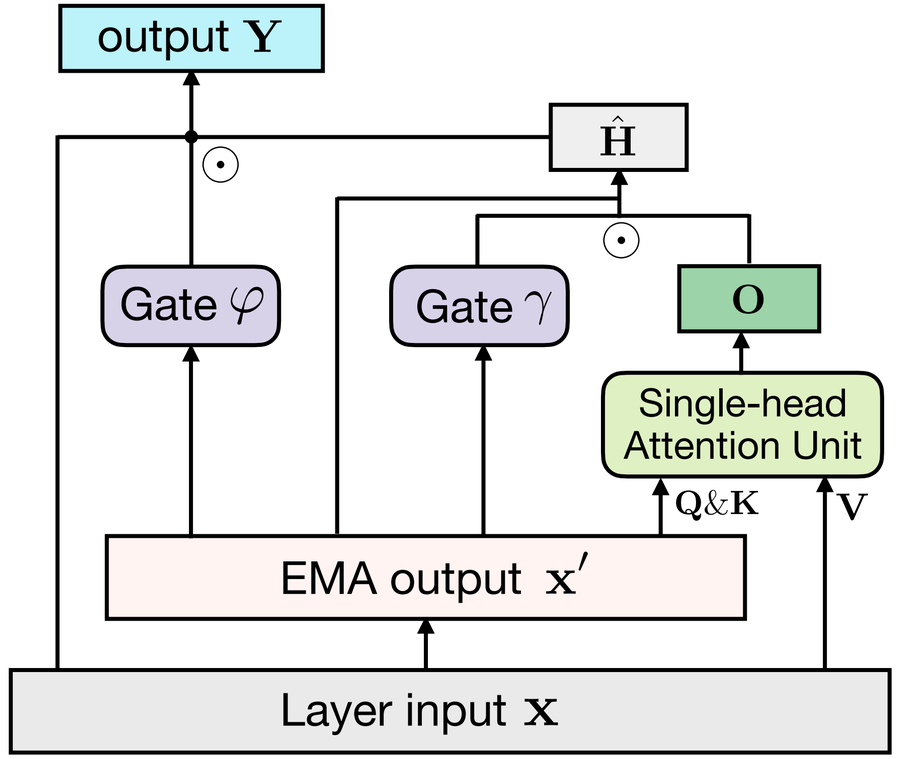
Theorem: *Suppose the transformation $\mathcal{G}$ is a universal function approximator.*

Then, for each $\boldsymbol{X}$ there exists $\boldsymbol{\gamma} = \mathcal{G}(\boldsymbol{X})$ such that $\boldsymbol{O}_{\text{SHGA}} = \boldsymbol{O}_{\text{MHA}}$

Mega Architecture: Outline

- **Exponential Moving Average (EMA)**
  - Local dependencies that decaying exponentially over time
  - Connection with State Space Models (e.g S4)

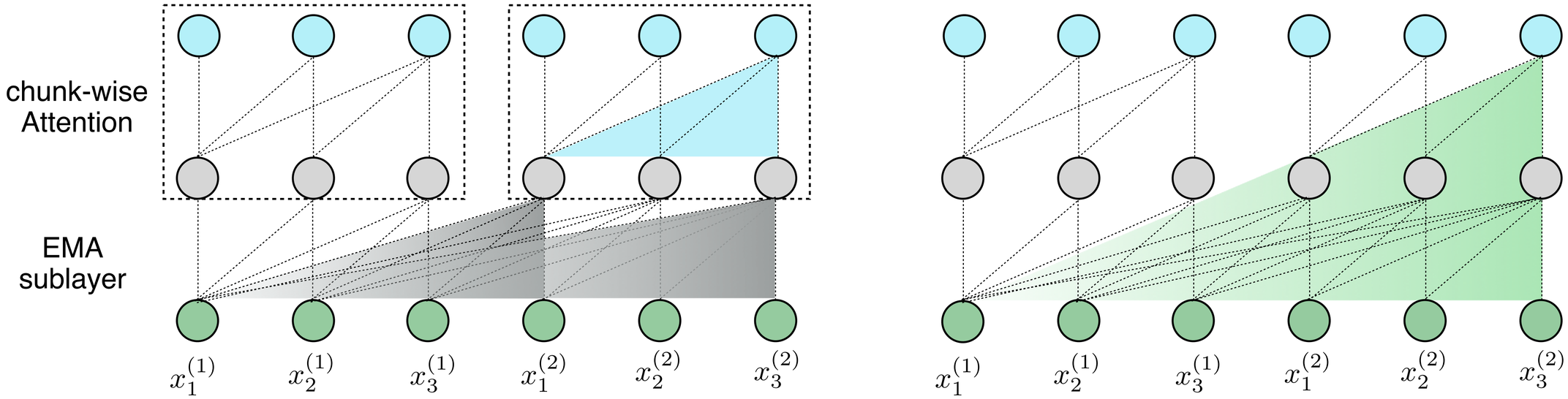
- **Single-head Gated Attention**
  - Adding a reset gate to the attention output
  - Theoretically proving that single-head gated attention is as expressive as multi-head one



- **Mega-Chunk**
  - Applying attention to local chunks of fixed length
  - Reducing quadratic complexity to linear

Mega-Chunk: Efficient Mega

- Split input sequences into multiple chunks with fixed length
- Applying attention individually to each chunk
  - Linear complexity & easy implementation
  - Losing contextual information between chunks
  - Fortunately, EMA preserves the information from previous chunks



Experiments

- **Long Range Arena (LRA)**
  - 3 tasks on byte-level text classification
  - 3 tasks on pixel-level image classification
- **Language Modeling**
  - Enwiki8 (character-level)
  - WikiText-103 (Word-level)
- **Machine Translation**
  - WMT'14 English - German
- **Image Classification**
  - ImageNet-1K
- **Raw Speech Classification**
  - Speech Commands

Experimental Results

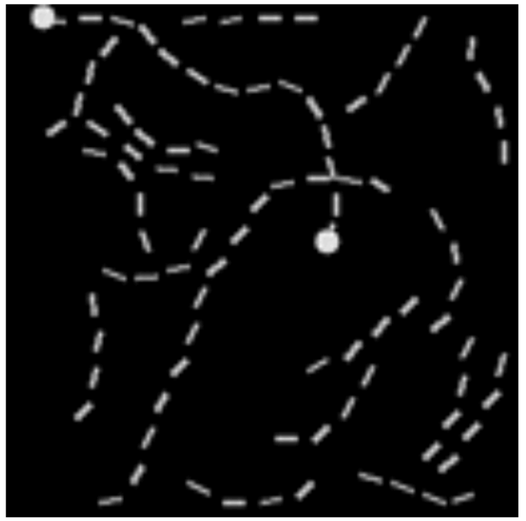
	LRA	WMT'14	WikiText-103	ImageNet	Raw-SC
XFM	59.24	27.68	18.66	81.80	31.24
S4	85.86	—	20.95	—	**97.50**
Mega	**88.21**	**29.01**	**18.07**	**82.31**	97.30

Analysis on LRA: Accuracy & Efficiency

- **A benchmark of 6 tasks for long-range sequence modeling**
  - Intentionally designed to be challenging
- **Tasks**
  - Long ListOpts:

    INPUT: `[MAX 4 3 [MIN 2 3 ] 1 0 [MEDIAN 1 5 8 9, 2]]` **OUTPUT:** `5`

  - Byte-level text classification on IMDB (> 4k tokens)
  - Byte-level document retrieval on ACL Anthology Network (ANN) (> 2k tokens)
  - Pixel-level image classification on CIFAR-10 (32x32 = 1024 token
  - PathFinder (32x32 = 1024 tokens)
  - Path-X (Extreme Length: 128x128 = 16,384 tokens)



Analysis on LRA: Accuracy & Efficiency

Models	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg.	Speed	Mem.
XFM	36.37	64.27	57.46	42.44	71.40	✗	54.39	–	–
XFM‡	37.11	65.21	79.14	42.94	71.83	✗	59.24	1×	1×
Reformer	37.27	56.10	53.40	38.07	68.50	✗	50.67	0.8×	0.24×
Linformer	35.70	53.94	52.27	38.56	76.34	✗	51.36	5.5×	0.10×
BigBird	36.05	64.02	59.29	40.83	74.87	✗	55.01	1.1×	0.30×
Performer	18.01	65.40	53.82	42.77	77.05	✗	51.41	**5.7**×	**0.11**×
Luna-256	37.98	65.78	79.56	47.86	78.55	✗	61.95	4.9×	0.16×
S4-v1	58.35	76.02	87.09	87.26	86.05	88.10	80.48	–	–
S4-v2	59.60	86.82	90.90	88.65	94.20	96.35	86.09	–	–
S4-v2‡	59.10	86.53	90.94	88.48	94.01	96.07	85.86	4.8×	0.14×
MEGA	**63.14**	**90.43**	**91.25**	**90.44**	**96.01**	**97.98**	**88.21**	2.9×	0.31×
MEGA-chunk	58.76	90.19	90.97	85.80	94.41	93.81	85.66	5.5×	0.13×

Limitations of Mega

- **Mode Capacity vs. Full Attention**
  - Limited capacity of EMA sub-layer
    - Mega-chunk fails behind Mega w. full attention

Analysis on LRA: Accuracy & Efficiency

Models	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg.	Speed	Mem.
XFM	36.37	64.27	57.46	42.44	71.40	✗	54.39	–	–
XFM‡	37.11	65.21	79.14	42.94	71.83	✗	59.24	1×	1×
Reformer	37.27	56.10	53.40	38.07	68.50	✗	50.67	0.8×	0.24×
Linformer	35.70	53.94	52.27	38.56	76.34	✗	51.36	5.5×	0.10×
BigBird	36.05	64.02	59.29	40.83	74.87	✗	55.01	1.1×	0.30×
Performer	18.01	65.40	53.82	42.77	77.05	✗	51.41	**5.7**×	**0.11**×
Luna-256	37.98	65.78	79.56	47.86	78.55	✗	61.95	4.9×	0.16×
S4-v1	58.35	76.02	87.09	87.26	86.05	88.10	80.48	–	–
S4-v2	59.60	86.82	90.90	88.65	94.20	96.35	86.09	–	–
S4-v2‡	59.10	86.53	90.94	88.48	94.01	96.07	85.86	4.8×	0.14×
MEGA	**63.14**	**90.43**	**91.25**	**90.44**	**96.01**	**97.98**	**88.21**	2.9×	0.31×
MEGA-chunk	58.76	90.19	90.97	85.80	94.41	93.81	85.66	5.5×	0.13×

Limitations of Mega

- **Mode Capacity vs. Full Attention**
  - Limited capacity of EMA sub-layer
    - Mega-chunk fails behind Mega w. full attention
- **Architectural Divergence on Difference Data Modalities**
  - Not only for Mega, but for almost all architectures
  - Different normalization layers on different positions
    - Batch Norm vs. Layer Norm vs. RMS Norm vs Scale Norm vs. …
    - Pre-Norm vs. Post-Norm vs. QK-Norm vs. …
  - Different attention functions
    - Softmax vs. ReLU2 vs. Laplace vs …
  - …

Analysis on LRA: Accuracy & Efficiency

Models	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg.	Speed	Mem.
XFM	36.37	64.27	57.46	42.44	71.40	✗	54.39	–	–
XFM‡	37.11	65.21	79.14	42.94	71.83	✗	59.24	1×	1×
Reformer	37.27	56.10	53.40	38.07	68.50	✗	50.67	0.8×	0.24×
Linformer	35.70	53.94	52.27	38.56	76.34	✗	51.36	5.5×	0.10×
BigBird	36.05	64.02	59.29	40.83	74.87	✗	55.01	1.1×	0.30×
Performer	18.01	65.40	53.82	42.77	77.05	✗	51.41	**5.7**×	**0.11**×
Luna-256	37.98	65.78	79.56	47.86	78.55	✗	61.95	4.9×	0.16×
S4-v1	58.35	76.02	87.09	87.26	86.05	88.10	80.48	–	–
S4-v2	59.60	86.82	90.90	88.65	94.20	96.35	86.09	–	–
S4-v2‡	59.10	86.53	90.94	88.48	94.01	96.07	85.86	4.8×	0.14×
MEGA	**63.14**	**90.43**	**91.25**	**90.44**	**96.01**	**97.98**	**88.21**	2.9×	0.31×
MEGA-chunk	58.76	90.19	90.97	85.80	94.41	93.81	85.66	5.5×	0.13×

Unified Architecture for Multi-Modality

- **Why a Unified Architecture Important for Multi-modality?**
  - A unified foundation model learning from various types of signals

We want a unified architecture that works well across different modalities!

Limitations of Mega

- **Mode Capacity vs. Full Attention**
  - Limited capacity of EMA sub-layer
    - Mega-chunk fails behind Mega w. full attention
- **Architectural Divergence on Difference Data Modalities**
  - Not only for Mega, but for almost all architectures
  - Different normalization layers and positions
    - Batch Norm vs. Layer Norm vs. RMS Norm vs Scale Norm vs. …
    - Pre-Norm vs. Post-Norm vs. QK-Norm vs. …
  - Different attention functions
    - Softmax vs. ReLU2 vs. Laplace vs. …
  - …
- **No Evidences or Results for Mega's Scalability**
  - Large-scale pretraining with Mega?

Megalodon: An Improved Version of Mega

- **Complex Exponential Moving Average**
  - Extending EMA to the Complex Field
- **Timestep Normalization**
  - Auto-regressive group normalization across sequential dimension
- **Numerical Stability**
  - Normalized Attention
  - Pre-Norm w. Top-hop Residual

Complex EMA

- **Directly inspired from state space models (SSMs)**
  - Almost all matrices diagonalizes over the complex plane
- **Extending EMA to the Complex Field**

$$\mathbf{y}_t = \boldsymbol{\alpha} \odot \mathbf{x}_t + (1 - \boldsymbol{\alpha} \odot \boldsymbol{\delta})(\cos\theta + i\sin\theta) \odot \mathbf{y}_{t-1}$$

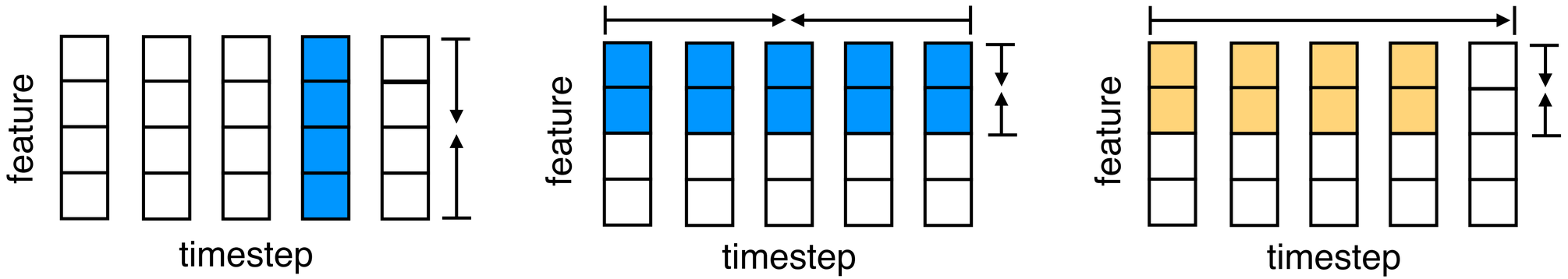
complex term in polar form

Still preserving the decaying structure in kernel weights

Time-step Normalization

- **Auto-regressive Group Normalization**
  - Allowing to normalize across the sequential/spatial dimension
  - Reducing internal covariate shift (same as Batch & Group Normalization)
- **Fused Cuda Implementation**
  - Efficiently calculating the cumulative mean and variance

$$\mu_t = \frac{1}{t}\sum_{j=1}^{t} x_j \qquad \sigma_t^2 = \frac{1}{t}\sum_{j=1}^{t} (x_j - \mu_t)^2$$



(a) Layer Norm (b) Group Norm (c) Timestep Norm

Numerical Stability

- **Normalized Attention**
  - Normalized attention mechanism: simplified version of Mega attention

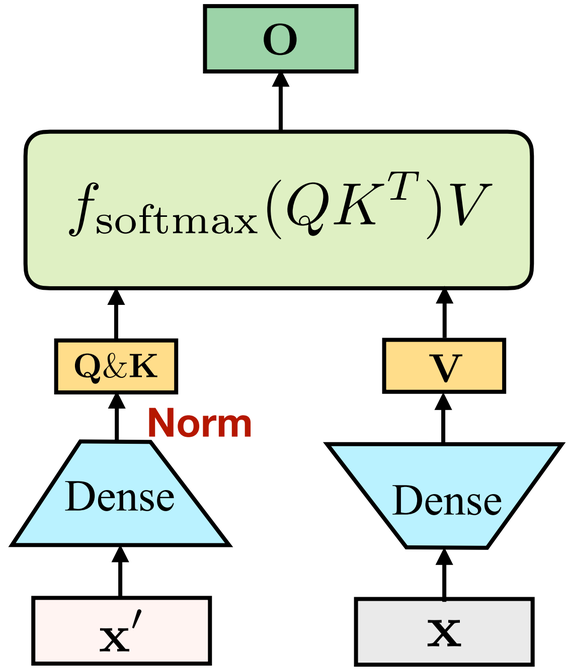
$$\boldsymbol{X}' = \mathrm{CEMA}(\boldsymbol{X}) \in \mathbb{R}^{n \times d}$$

$$\boldsymbol{Z} = \boldsymbol{X}' W_z + b_z, \quad \boldsymbol{Z}' = \frac{\boldsymbol{Z}}{\|\boldsymbol{Z}\|} \in \mathbb{R}^{n \times z}$$

$$\boldsymbol{Q} = \boldsymbol{\kappa}_q \odot \boldsymbol{Z}' + \boldsymbol{\mu}_q \in \mathbb{R}^{n \times z}$$

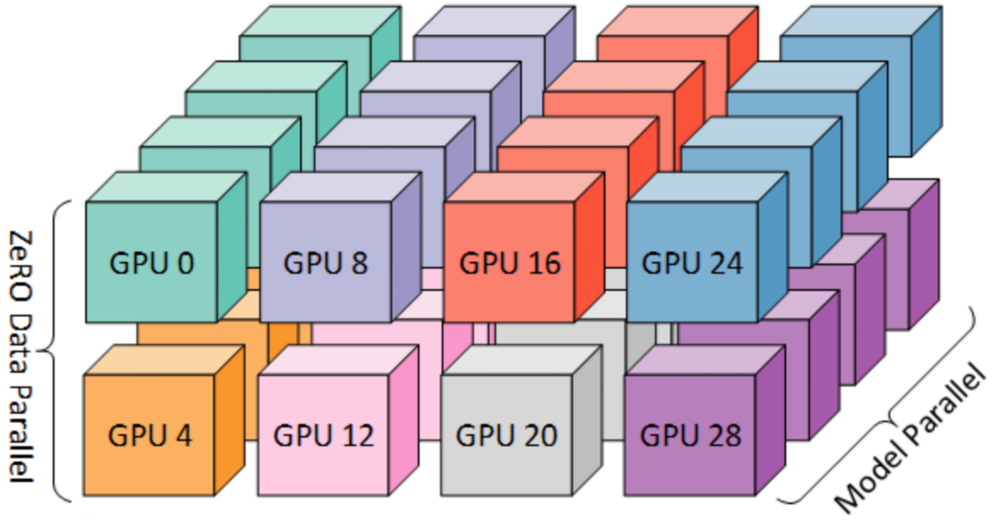
$$\boldsymbol{K} = \boldsymbol{\kappa}_k \odot \boldsymbol{Z}' + \boldsymbol{\mu}_k \in \mathbb{R}^{n \times z}$$

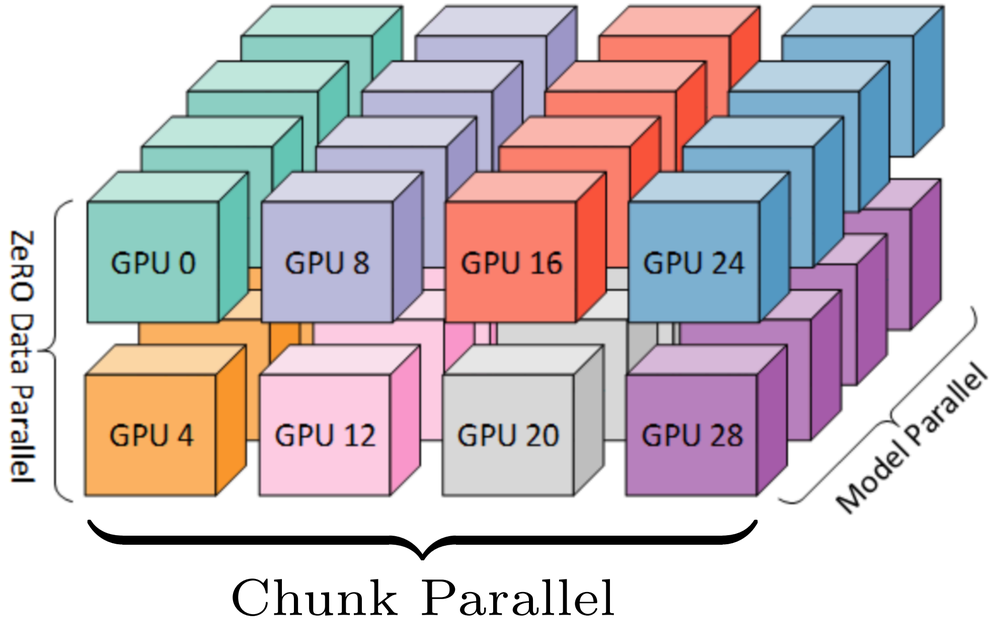
$$\boldsymbol{O} = f_{\mathrm{softmax}}\left(\boldsymbol{Q}\boldsymbol{K}^T\right)\boldsymbol{V} \in \mathbb{R}^{n \times v}$$



Revisiting Quadratic Complexity in Attention

- **It is hard to do distributed training across the sequence dimension on multiple devices with Transformers**

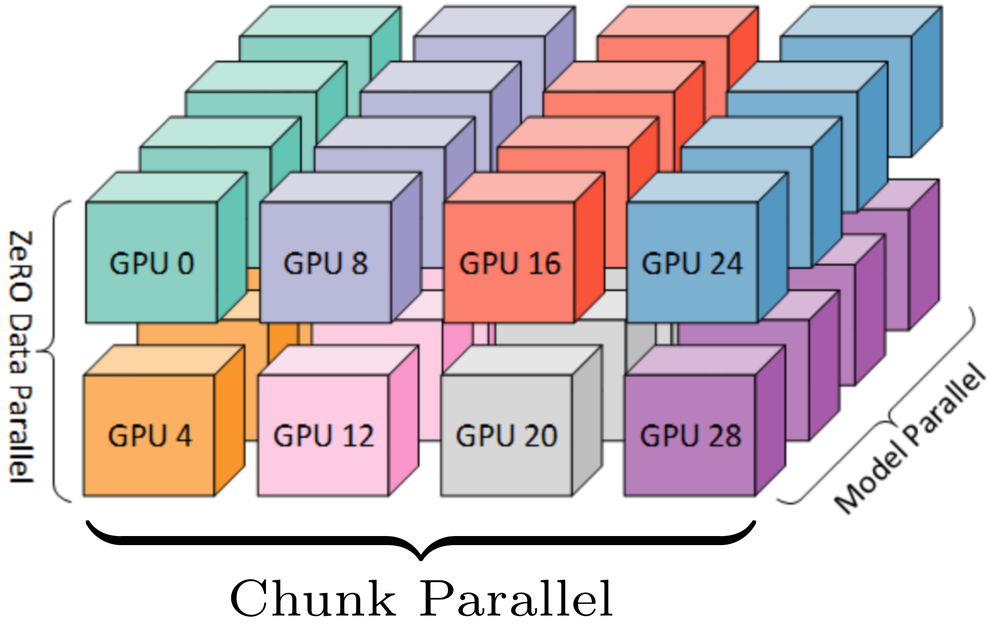




Megalodon introduces chunk parallel

Image modified from Microsoft blog

Chunk Parallel



Megalodon introduces chunk parallel

- Communication (point-wise) is cheap:
  - last hidden states of CEMA
  - cum mean and var of timestep norm
- Minimize communication cost:
  - hide communication costs in computation

Image modified from Microsoft blog

Megalodon: Results on Small-Scale Benchmarks

	LRA	SC	ImageNet	Enwiki8	WT103
S4	85.86	97.50	—	—	20.95
XFM	59.24	xx	81.8	1.08	18.66
Mega-chunk	85.66	96.92	—	1.02	18.07
Mega	88.21	—	82.35	—	—
Megalodon-chunk	**87.62**	**98.14**	**—**	**1.00**	**17.23**
Megalodon	**88.63**	**—**	**83.12**	**—**	**—**

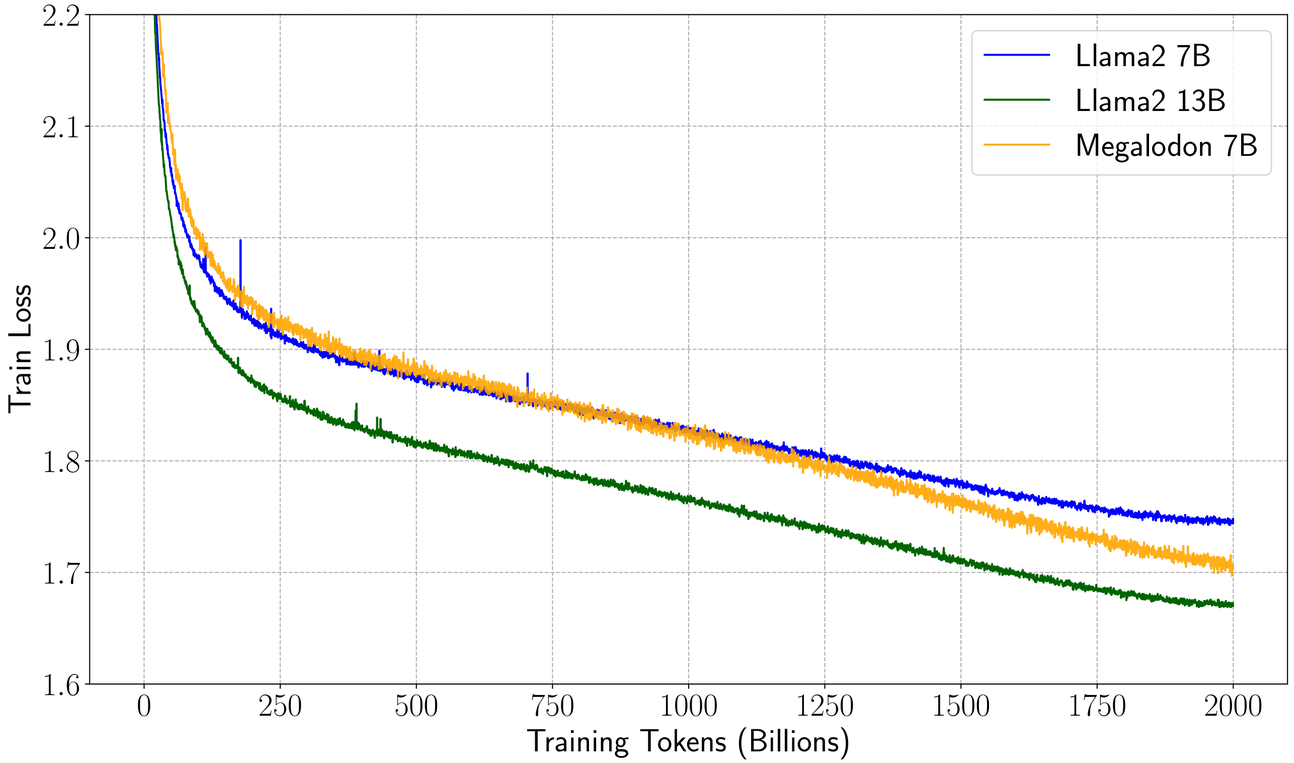
Importantly! All the experiments were conducted using exactly the same architecture!

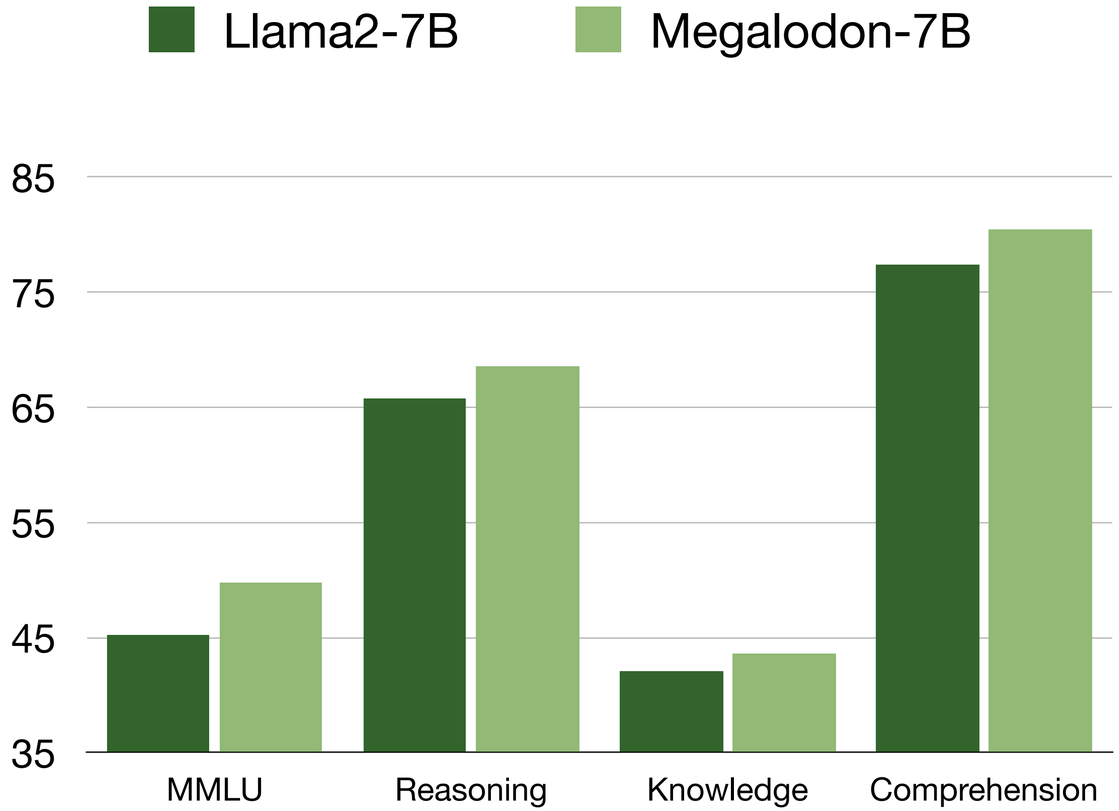
Megalodon on LLM Pretraining

- **Pretraining Data**
  - 2 trillion tokens
  - Exactly the same with Llama2 for head-to-head comparison
- **Architecture Hyperparameters**
  - Closely following Llama2
    - 7B parameters
    - 32 blocks, model dimension 4096
    - Rotary positional embedding (RoPE)
  - Differences
    - 4 attention heads (32 in Llama2)
    - 32K context length w. 4K attention chunk size (4K full attention in Llama2)

Efficiency of Megalodon-7B

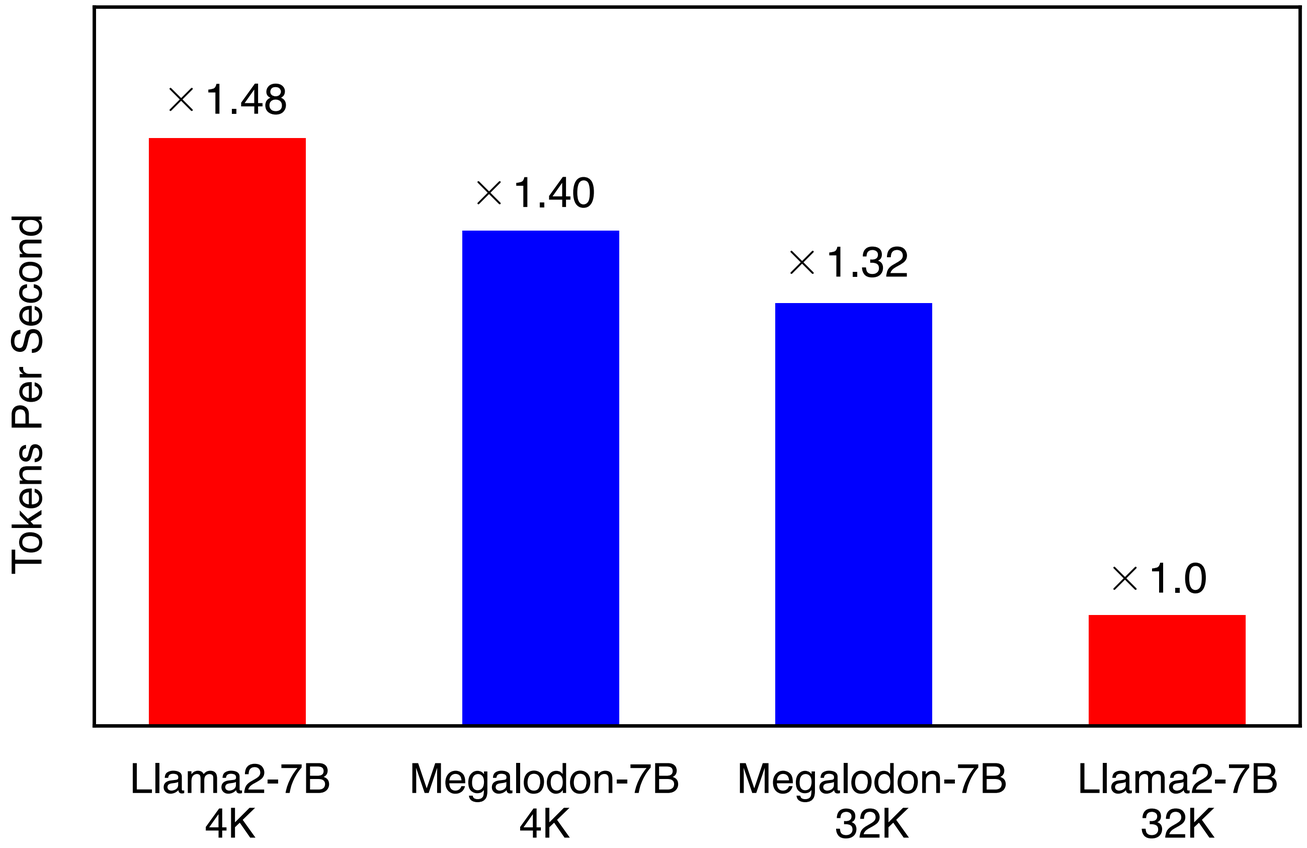
- **Data Efficiency**





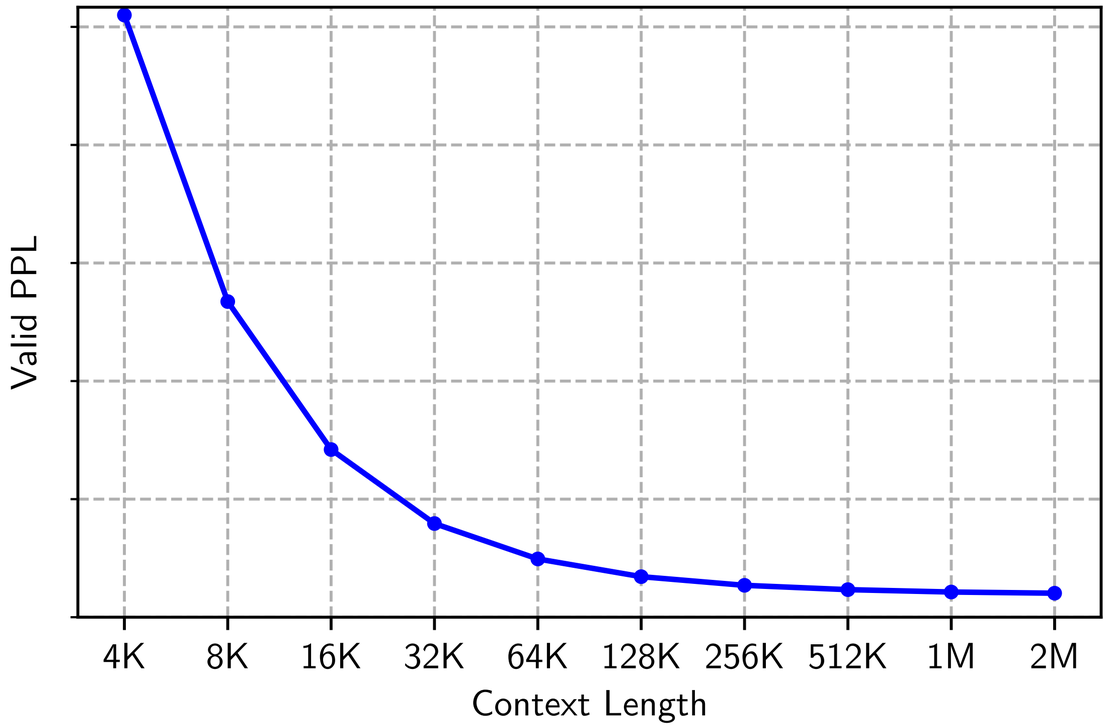
Efficiency of Megalodon-7B

- Compute Efficiency



Long-Context Modeling

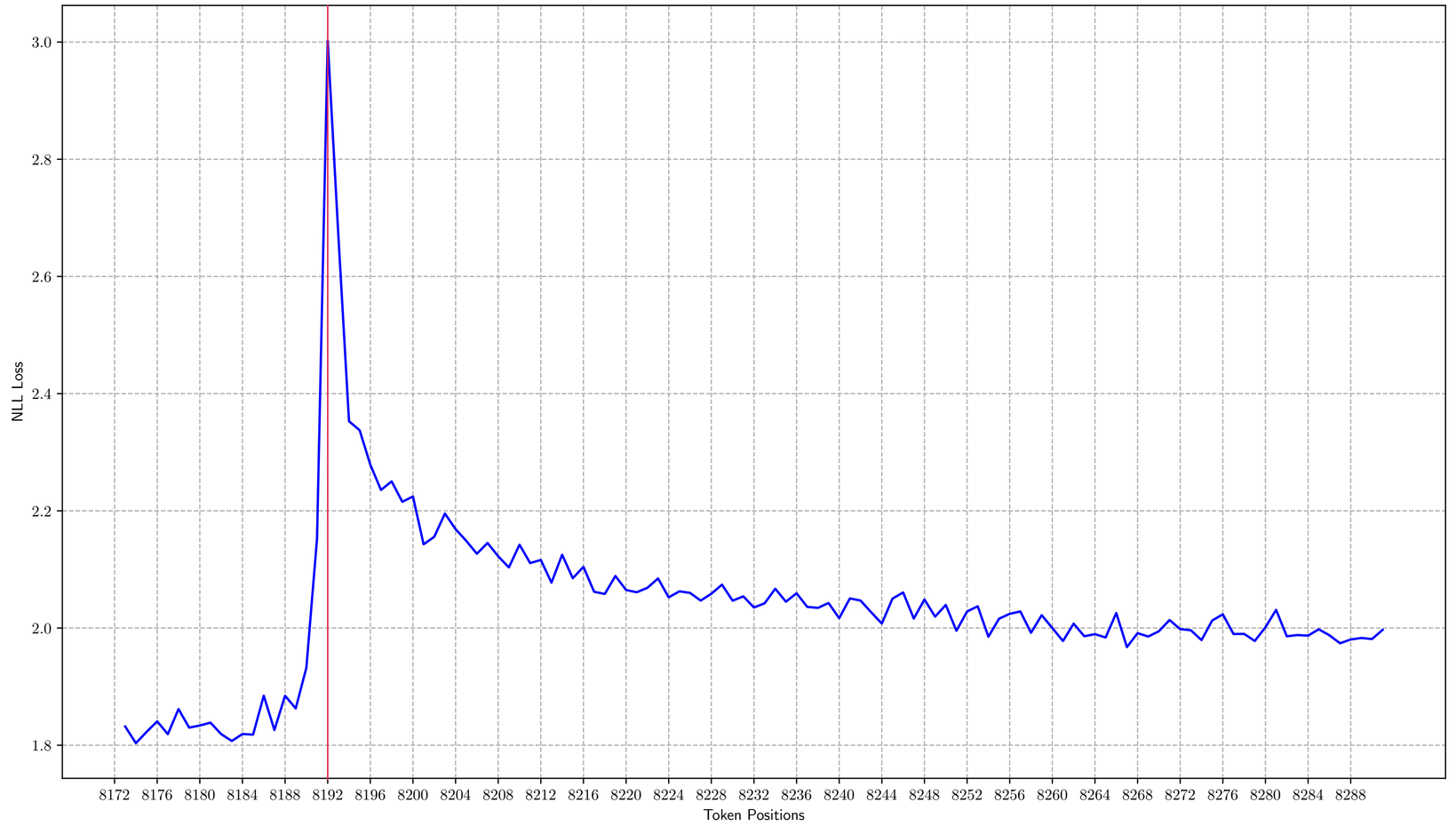
- **Long-Context Evaluation of Megalodon-7B**
  - Perplexity w. various context lengths
  - Long-context QA tasks in Scrolls



Model	NaQA	Qasper	QMSum
Xgen	17.4	20.5	6.8
MPT	18.8	24.7	8.8
Yarn	20.9	26.2	11.4
LLAMA2	18.8	19.8	10.1
LLAMA2-L*	23.5	**28.3**	**14.5**
MEGALODON	**23.9**	28.0	13.1

Existing Problems of Megalodon

- Validation losses on positions before and after chunk boundaries increase significantly

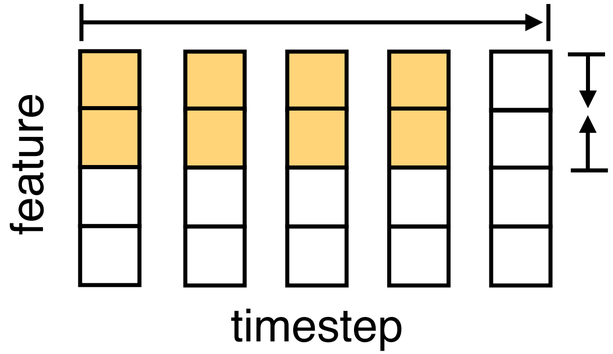


Gecko: An Improved Version of Mega

- **Inherent Capability of Length Generalization**
  - Inherently processing sequences with unlimited contexts
    - No length extension
  - Better capability of in-context learning
  - Three new components:
    - Timestep Normalization -> Timestep Decay Normalization
    - Chunk-wise attention -> Sliding chunk attention
    - Adaptive working memory via online softmax linear attention

Timestep Normalization

$$\mu_t = \frac{1}{d}\sum_{j=1}^{d} x_{t,j} \qquad \sigma_t^2 = \frac{1}{d}\sum_{j=1}^{d}(x_{t,j} - \mu_t)^2$$



(c) Timestep Norm

Running Mean and Var

$$m_t = \frac{1}{t}\sum_{i=1}^{t} \mu_i \qquad v_t = \frac{1}{t}\sum_{i=1}^{t}\frac{1}{d}\sum_{j=1}^{d}(x_{i,j} - m_t)^2$$

Timestep Decay Normalization

$$\mu_t = \frac{1}{d}\sum_{j=1}^{d} x_{t,j} \qquad \sigma_t^2 = \frac{1}{d}\sum_{j=1}^{d} (x_{t,j} - \mu_t)^2$$

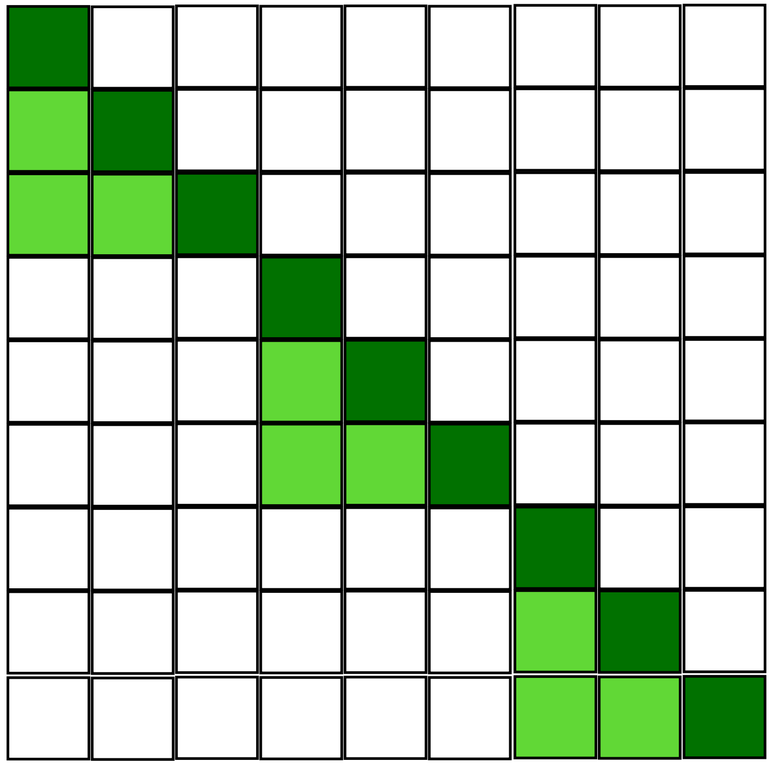
Running Mean and Var via decay

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\mu_t \qquad v_t = \beta_2 v_{t-1} + (1-\beta_2)\sigma_t^2$$

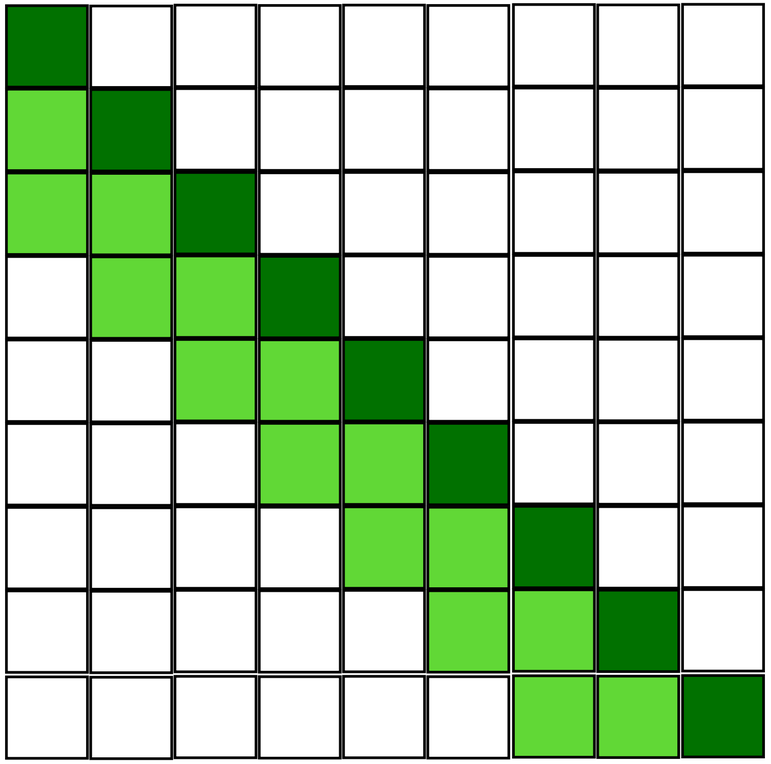
Bias Correction

$$m_t' = \frac{m_t}{1-\beta_1^t} \qquad v_t' = \frac{v_t}{1-\beta_2^t}$$

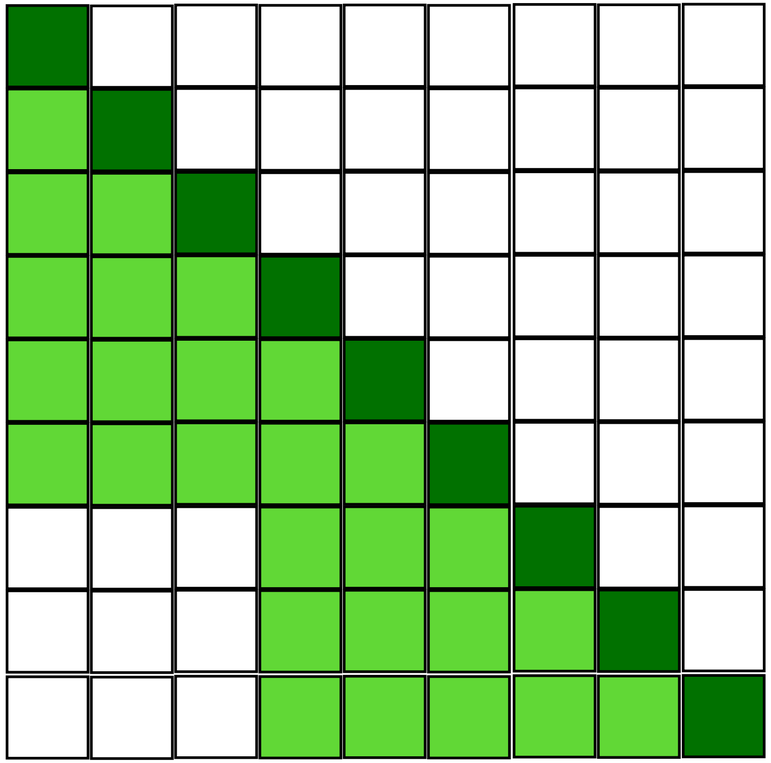
Sliding Chunk Attention



Chunk-wise Attention



Sliding Window Attention



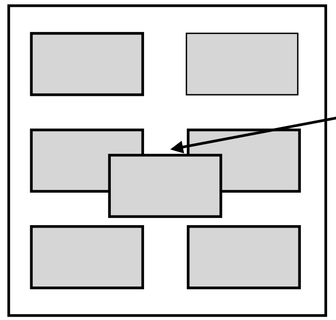
Sliding Chunk Attention

Adaptive Working Memory

Working Memory via Linear Attention

$$\mathbf{M}_s = \mathbf{M}_{s-1} + \phi(K_s)^T V_s$$

position-independent

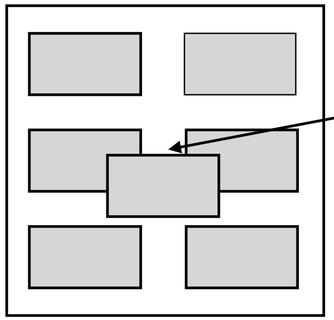


Adaptive Working Memory

Working Memory via Linear Attention

$$\mathbf{M}_s = \mathbf{M}_{s-1} + \phi(K_s)^T V_s$$

position-independent



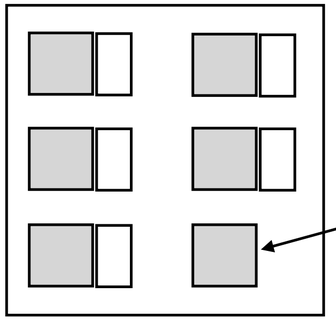
Adaptive Working Memory

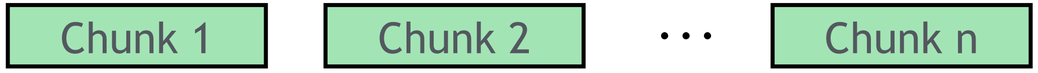
$$\mathbf{M}_s = \frac{z_{s-1}}{z_s} \odot \mathbf{M}_{s-1} + \phi_s(K_s)^T V_s$$

$$\phi_s(\mathbf{K}_s) = \frac{\exp(\mathbf{K}_s)}{z_s}$$

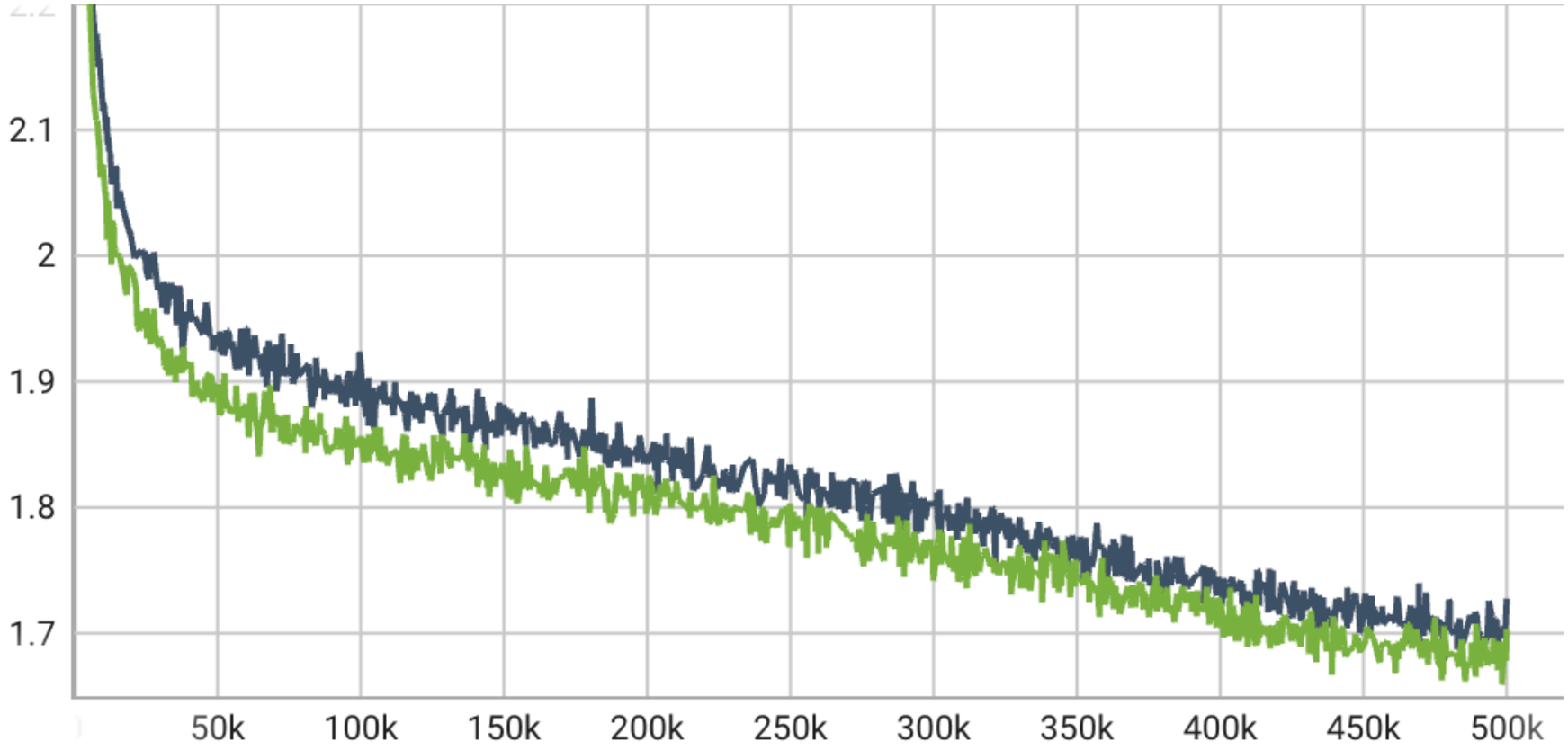
position-dependent

$$z_s = \sum_{c=1}^{s} \exp(\mathbf{K}_c)$$



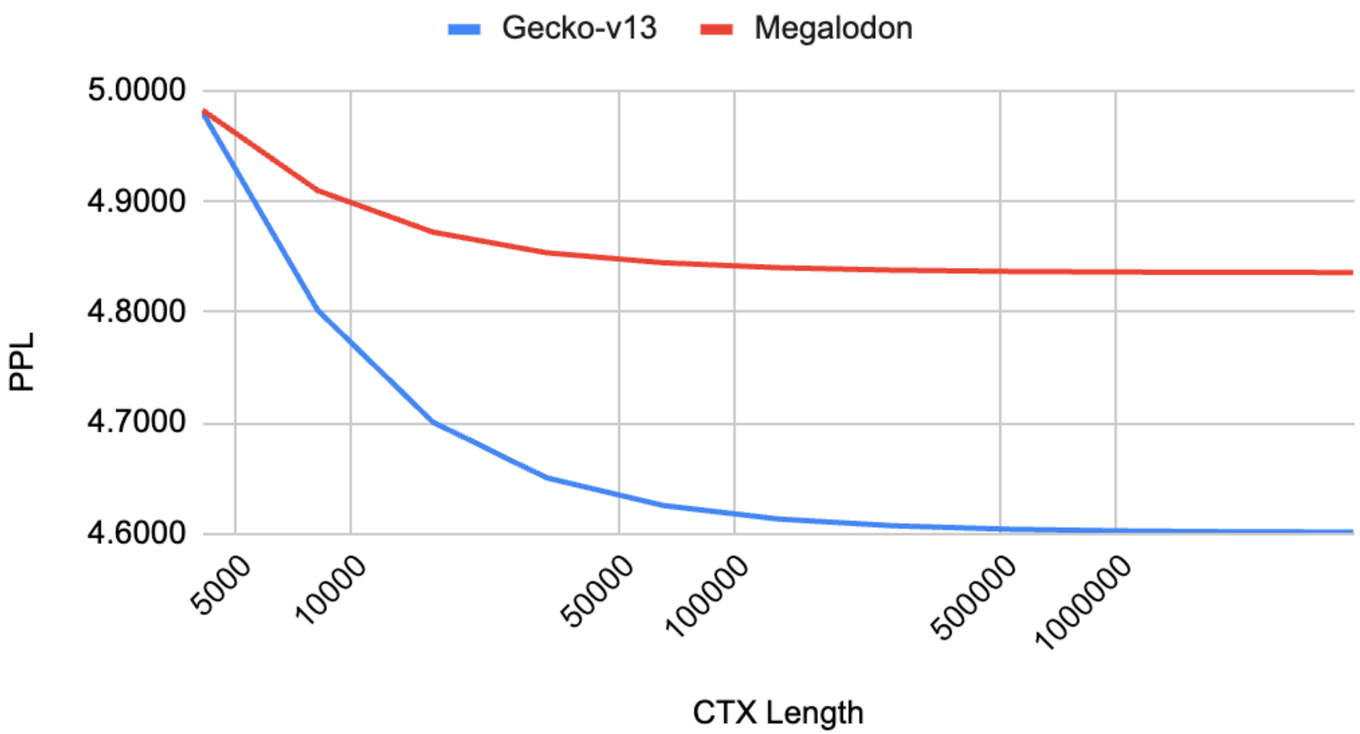


Pre-training Loss



Long-Context Modeling

- **Long-Context Evaluation (7B)**
  - Perplexity w. various context lengths
  - Long-context QA tasks in Scrolls



	NaQA	Qasper	QMSum
Llama2	18.8	19.8	10.1
Llama2-L	23.5	28.3	14.5
Megalodon	23.9	28.0	13.1
Gecko	**26.4**	**31.8**	**15.4**

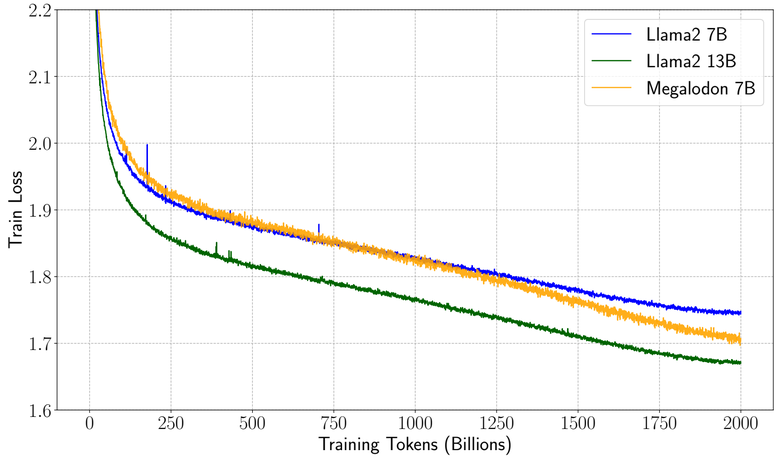
Thanks!
Q&A

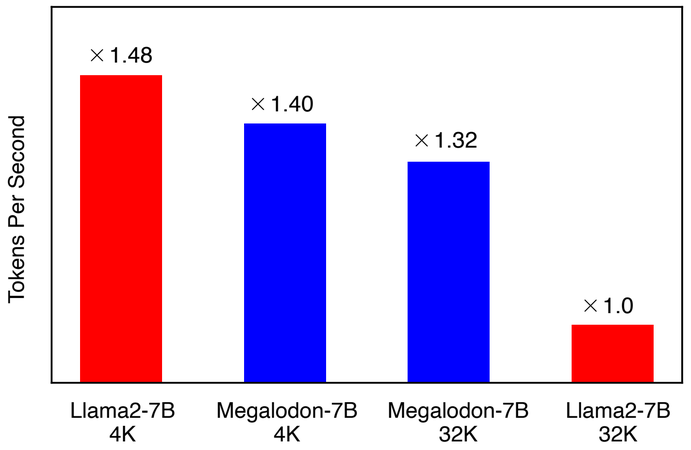
Code: https://github.com/XuezheMax/megalodon

Megalodon: Efficient LLM with Unlimited Context Length

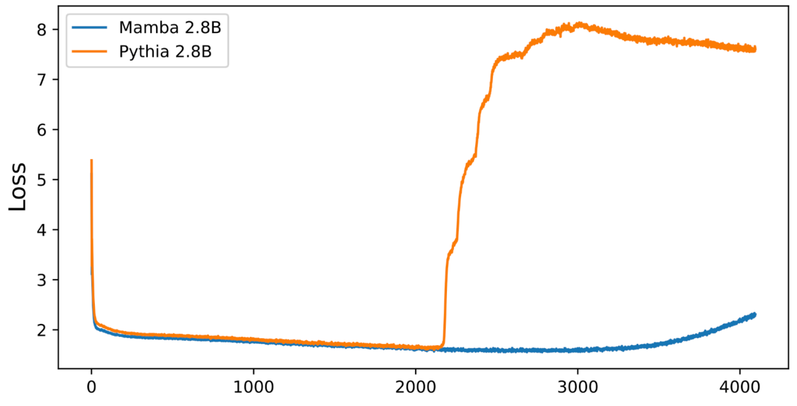
- **Efficient unified model architecture beyond Transformer for long-sequential data from various modalities**

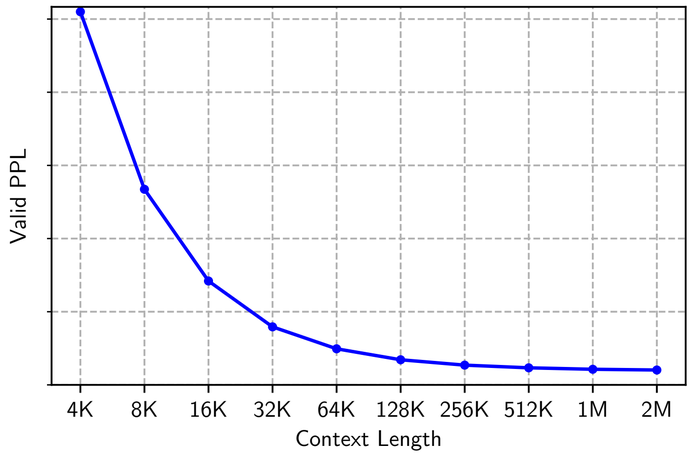
Efficiency vs. LlaMa2





Long-Context Modeling





Unified Architecture

- A unified model architecture learning from various types of signals
- No architectural divergence on different modalities:
  - Different normalization layers and positions
    - Batch Norm vs. Layer Norm vs. RMS Norm vs. Group Norm vs. …
    - Pre-Norm vs. Post-Norm vs. QK-Norm vs. …
  - Different attention functions
    - Softmax vs. ReLU2 vs. Laplace vs. …
  - …