南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Advanced Natural Language Processing

Lecture 14: Transformer in the LLM Era

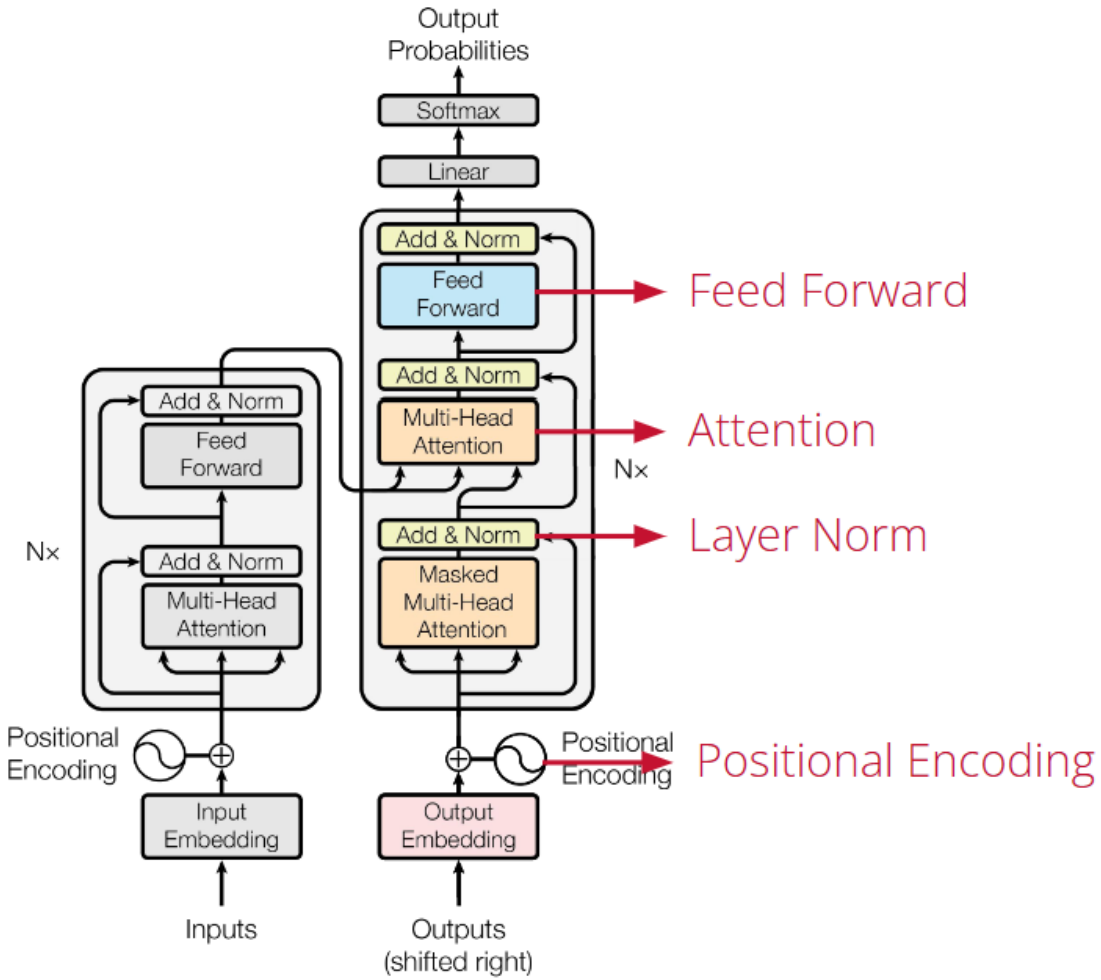
陈冠华 CHEN Guanhua

Department of Statistics and Data Science

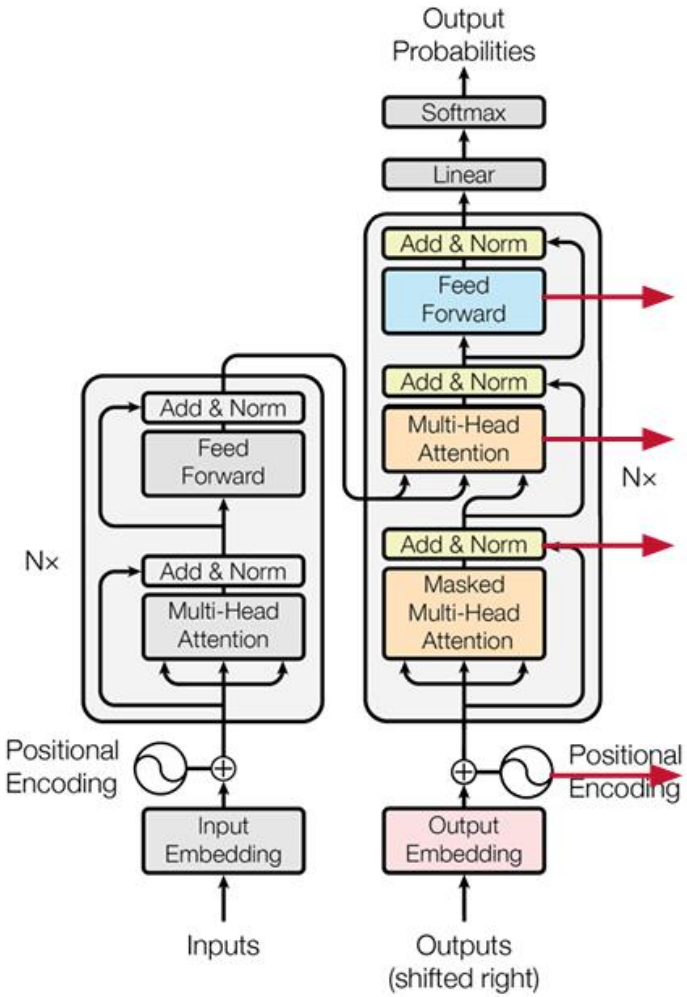
Content

- FFN Variants
- Attention Variants
- Layer Normalization Variants
- Positional Embedding Variants

Places for Improvements



LLaMA3’ Choice



Feed Forward: SwiGLU

Attention: Grouped-query Attention

Layer Norm: Pre-layernorm with RMSNorm

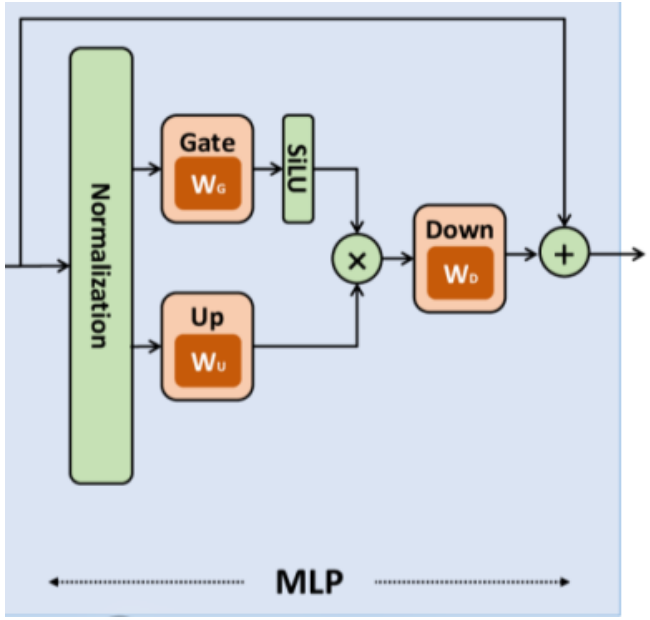
Positional Encoding: RoPE

Traditional vs. Advanced

Category	Original Setting	Improvement Setting
Positional Embedding	Absolute Position	Rotary Position (RoPE)
Layer Normalization	Post-LN, Layer-Norm	Pre-LN, RMS-Norm
Activation Function	ReLU	SWiGLU
Attention	Multi-Head Attention	Multi-Query Attention (MQA) **Group-Query Attention (GQA)** Multi-Head Latent Attention (MLA), Deepseek use it Sliding Window Attention (SWA), Mistral use GQA+SWA

Transformer Improvements汇总 - 飞书云文档

Feed-Forward Layer




```python
class FeedForward(nn.Module):
    def __init__(self, dim: int, hidden_dim: int, multiple_of: int,
dropout: float):
        super().__init__()
        hidden_dim = int(2 * hidden_dim / 3)
        hidden_dim = multiple_of * ((hidden_dim + multiple_of - 1) //
multiple_of)
        self.w1 = nn.Linear(dim, hidden_dim, bias=False)
        self.w2 = nn.Linear(hidden_dim, dim, bias=False)
        self.w3 = nn.Linear(dim, hidden_dim, bias=False)
        self.dropout = nn.Dropout(dropout)


    def forward(self, x):
        return self.dropout(self.w2(F.silu(self.w1(x)) * self.w3(x)))  #
SiLU(x) = x * sigmoid(x)
```

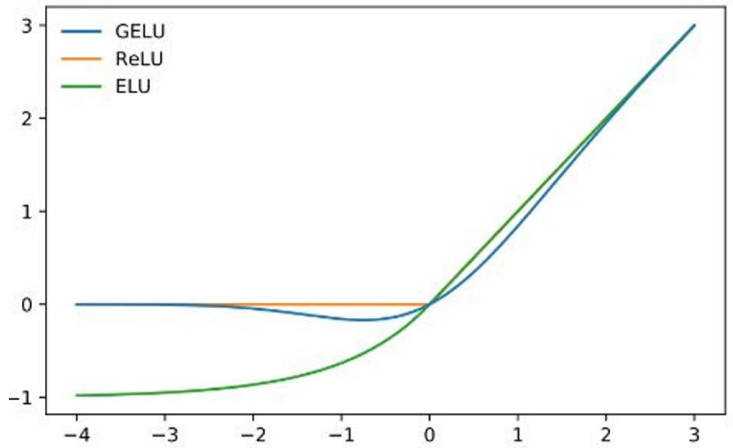
Feed Forward: Activations

Variants of linear FFN (omitting bias):

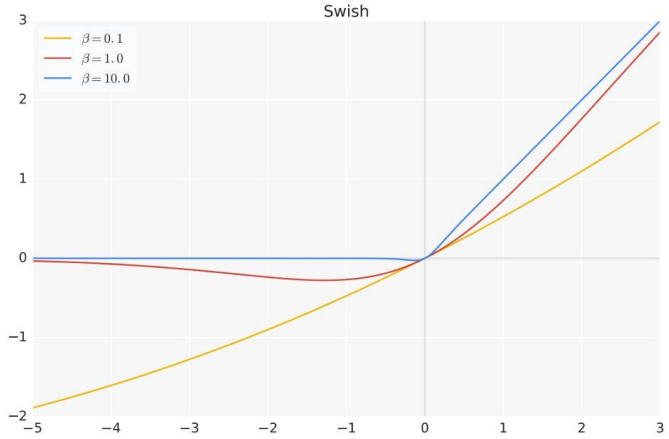
$$\mathrm{FFN}_{\mathrm{RELU}}(x) = \mathrm{RELU}(x\mathbf{W}_1)\mathbf{W}_2;\ \mathrm{RELU}(x\mathbf{W}_1) = \max(0, x\mathbf{W}_1)$$

$$\mathrm{FFN}_{\mathrm{GELU}}(x) = \mathrm{GELU}(x\mathbf{W}_1)\mathbf{W}_2; \mathrm{GELU}(x\mathbf{W}_1) = xP(X < x) = x\Phi(x)$$

$$\mathrm{FFN}_{\mathrm{Switch}}(x) = \mathrm{Swish}_1(x\mathbf{W}_1)\mathbf{W}_2; \mathrm{Swish}_\beta(x\mathbf{W}_1) = x\mathrm{Sigmod}(\beta x)$$



GELU Activation



Swish Activation

Feed Forward: Bilinear Layers

Bilinear FFNs (omitting bias):

$$\mathrm{FFN}_{\mathrm{Bilinear}}(x) = (x\mathbf{W} \cdot x\mathbf{V})\mathbf{W}_2. \text{ Two FFN with componentwise product}$$

$$\mathrm{FFN}_{\mathrm{ReGLU}}(x) = (\mathrm{RELU}(x\mathbf{W}) \cdot x\mathbf{V})\mathbf{W}_2. \text{ Adding RELU activation on one FFN}$$

$$\mathrm{FFN}_{\mathrm{GEGLU}}(x) = (\mathrm{GELU}(x\mathbf{W}) \cdot x\mathbf{V})\mathbf{W}_2. \text{ Adding GELU activation on one FFN}$$

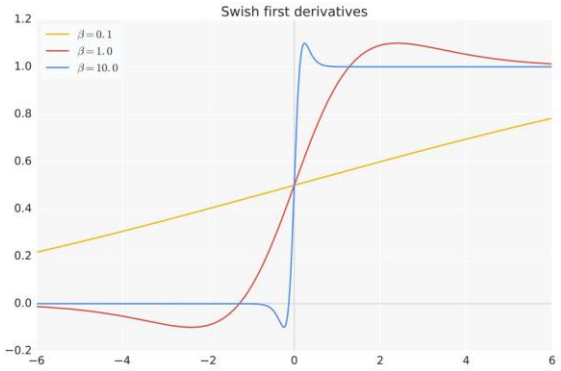
$$\mathrm{FFN}_{\mathrm{SwiGLU}}(x) = (\mathrm{Swish}_1(x\mathbf{W}) \cdot x\mathbf{V})\mathbf{W}_2. \text{ Adding Swish activation on one FFN}$$

With reduced hidden dimension of the projections to keep parameter count the same

Training Steps	65,536	524,288
$\mathrm{FFN}_{\mathrm{ReLU}}(baseline)$	1.997 (0.005)	1.677
$\mathrm{FFN}_{\mathrm{GELU}}$	1.983 (0.005)	1.679
$\mathrm{FFN}_{\mathrm{Swish}}$	1.994 (0.003)	1.683
$\mathrm{FFN}_{\mathrm{GLU}}$	1.982 (0.006)	1.663
$\mathrm{FFN}_{\mathrm{Bilinear}}$	1.960 (0.005)	1.648
$\mathrm{FFN}_{\mathrm{GEGLU}}$	**1.942** (0.004)	**1.633**
$\mathrm{FFN}_{\mathrm{SwiGLU}}$	**1.944** (0.010)	**1.636**
$\mathrm{FFN}_{\mathrm{ReGLU}}$	1.953 (0.003)	1.645

Improved speed-quality

T5 base Perplexity at Pretraining Steps



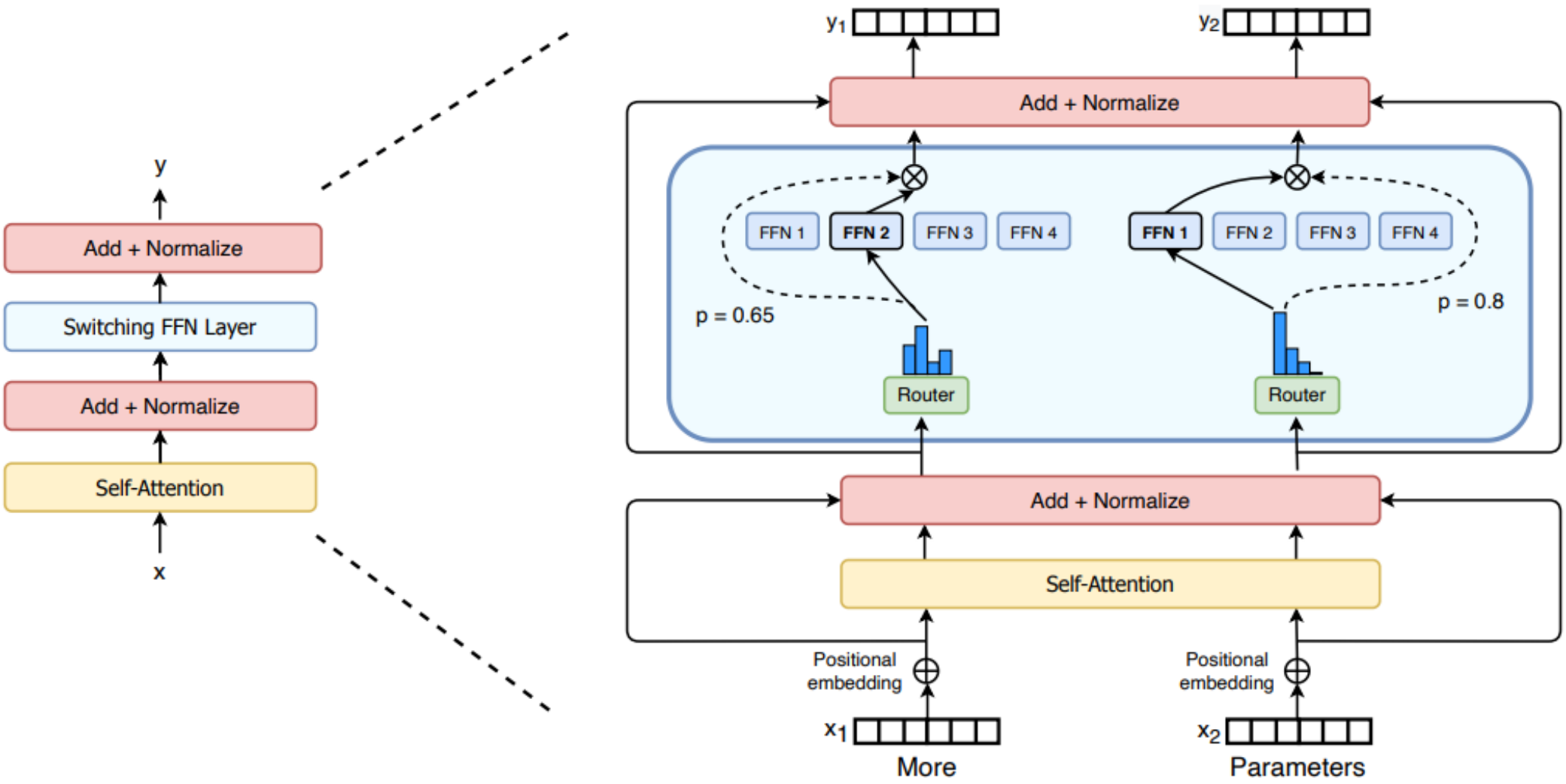
Swish Gradients

Mixture-of-Experts (MoE)

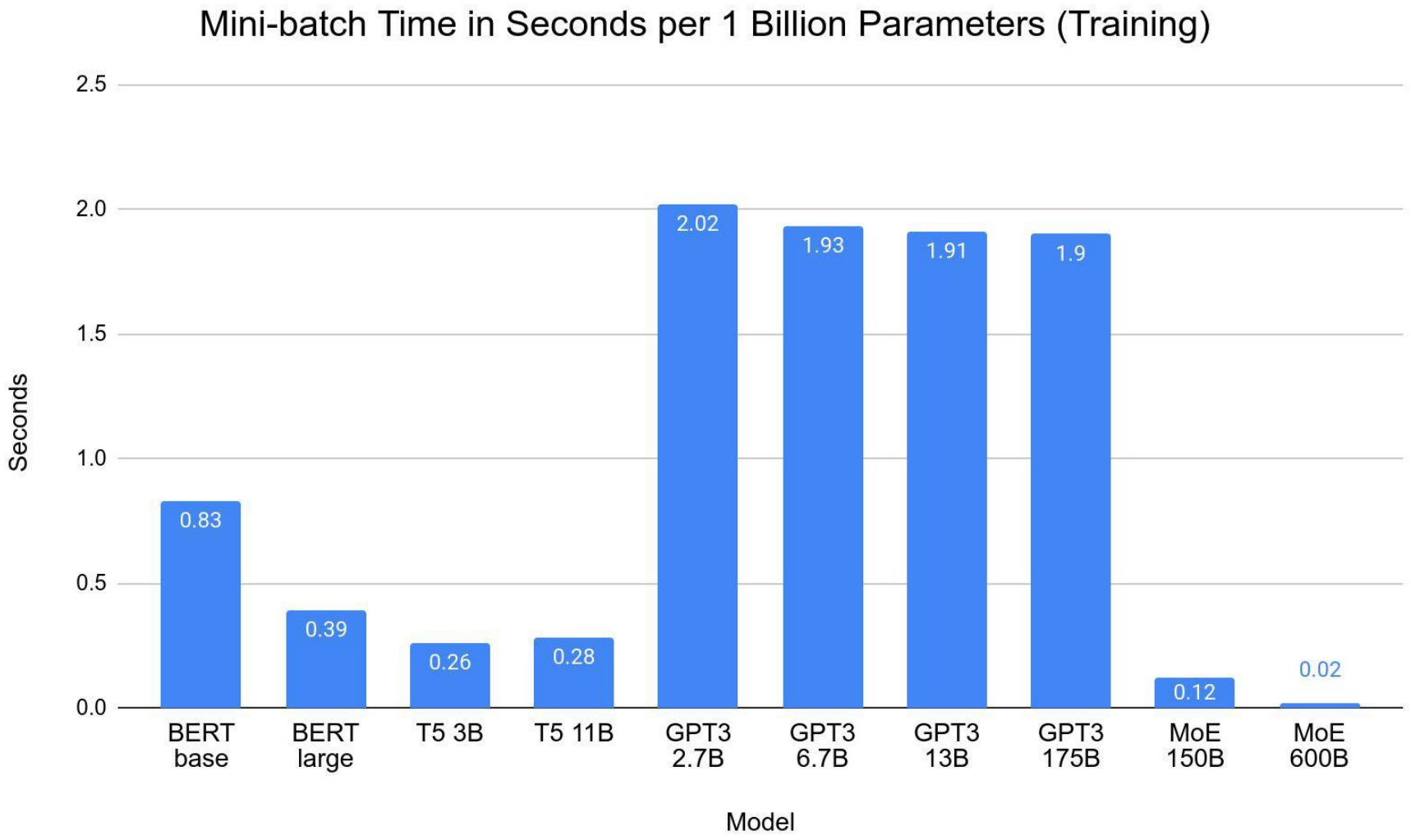
南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Within one deep neural network, ensembling can be implemented with a gating mechanism connecting multiple experts
- The gating mechanism controls which subset of the network (e.g. which experts) should be activated to produce outputs.
- One MoE layer contains
  - N feed-forward networks as experts
  - A trainable gating network G to learn a probability distribution over n experts so as to route the traffic to a few selected experts

Mixture-of-Experts (MoE)



Transformers Mini-batch Time



Gating Network

- A learned gating network (G) decides which experts (E) to send a part of the input

$$y = \sum_{i=1}^{n} G(x)_i E_i(x) \qquad G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$

- Noisy Top-K Gating mechanism
  - Introduces some (tunable) noise and then keeps the top k values (k=2/3)

$$H(x)_i = (x \cdot W_g)_i + \text{StandardNormal}() \cdot \text{Softplus}((x \cdot W_{\text{noise}})_i)$$

$$\text{KeepTopK}(v, k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v, \\ -\infty & \text{otherwise.} \end{cases}$$

$$G(x) = \text{Softmax}(\text{KeepTopK}(H(x), k))$$

Softplus — PyTorch 2.1 documentation

Load Balancing

- If all our tokens are sent to just a few popular experts, that will make training inefficient
  - The gating network converges to mostly activate the same few experts
- Auxiliary loss is added to encourage giving all experts equal importance
  - Ensures that all experts receive a roughly equal number of training examples

Random Routing

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- In a top-2 setup, we always pick the top expert
- But the second expert is picked stochastically

  - Compute the top expert: $e_1 = \arg\max_e g_e(x)$.
  - Compute the second expert: $e_2 = \arg\max_{e \neq e_1} g_e(x)$.
  - Always keep top expert and keep the second expert stochastically:
    - Let $p = \min(2g_{e_2}(x), 1)$.
    - With probability p, set $\tilde{g}_{e_1}(x) = \frac{g_{e_1}(x)}{g_{e_1}(x)+g_{e_2}(x)}$, $\tilde{g}_{e_2}(x) = \frac{g_{e_2}(x)}{g_{e_1}(x)+g_{e_2}(x)}$, $\tilde{g}_e(x) = 0$ for $e \notin \{e_1, e_2\}$.
    - With probability $1-p$: $\tilde{g}_{e_1}(x) = 1$, and $\tilde{g}_e(x) = 0$ for $e \neq e_1$.

Load Balancing

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Expert capacity
  - we can set a threshold of how many tokens can be processed by one expert
- Let c_e be the number of times expert e is selected, B be the number of tokens in the batch
  - **Auxiliary loss**: We would like to encourage $c = [c_1, \ldots, c_E]$ to close to uniform.
    - We could penalize $\|c\|_2^2 = \sum_{e=1}^{E} c_e^2$, but this is not differentiable.
    - Define $m_e = \sum_{i=1}^{B} g_e(x_i)$ (this is the soft version of c_e).
    - Instead, we add $\text{load-balancing-loss} = \sum_{e=1}^{E} m_e c_e$ to the objective function. This way, the gradient will be nonzero through m_e.

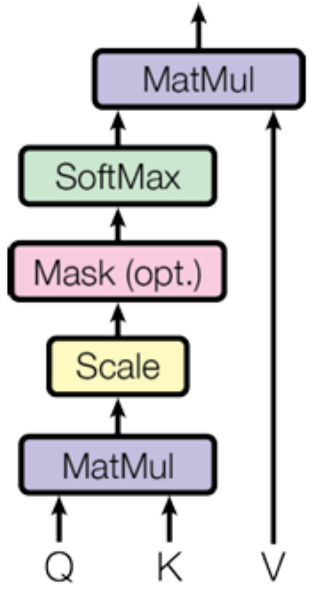
$$\text{loss} = \text{negative-log-likelihood} + \lambda \text{load-balancing-loss}.$$

For example, we can take $\lambda = \frac{0.01}{B}$.

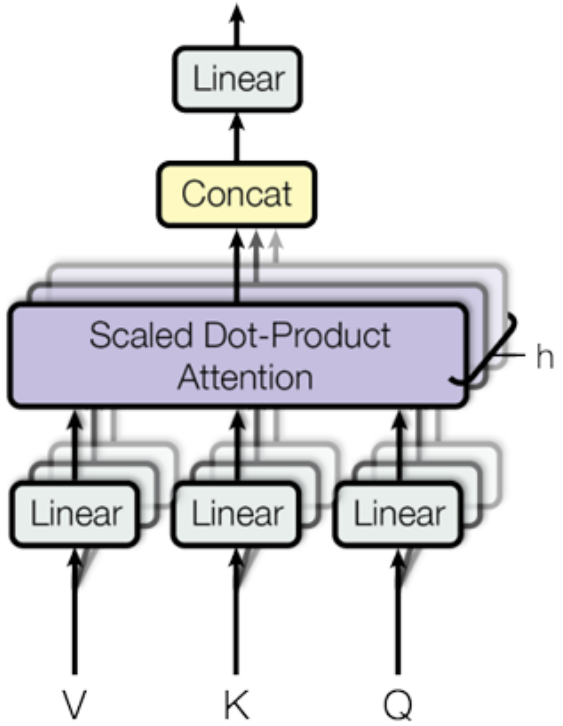
[2006.16668] GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding (arxiv.org)

Recap: Multi-Head Attention

Scaled Dot-Product Attention



Multi-Head Attention



$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Attention: Efficient Attention Mechanisms

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

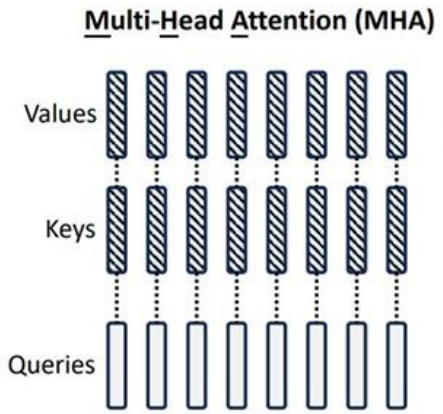
Standard Multi-head Attention

$$\text{head}_1 = \text{Attention}(\mathbf{Q}\mathbf{W}_1^Q, \mathbf{K}\mathbf{W}_1^K, \mathbf{V}\mathbf{W}_1^V)$$

$$\vdots$$

$$\text{head}_H = \text{Attention}(\mathbf{Q}\mathbf{W}_H^Q, \mathbf{K}\mathbf{W}_H^K, \mathbf{V}\mathbf{W}_H^V)$$

$$\text{MultiHeadAtt}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)$$



Inputs and outputs of each layer are the same dimensions:

$$\mathbf{Q} \in \mathbb{R}^{T \times d_{\text{model}}}$$
$$\mathbf{K} \in \mathbb{R}^{T \times d_{\text{model}}}$$
$$\mathbf{V} \in \mathbb{R}^{T \times d_{\text{model}}}$$

→ Huge memory consumption during inference. Needs to keep one K, V for each layer and each position

Attention: Efficient Attention Mechanisms

Grouped-Query Attention: Divide Q in G groups, and share K, V in the same group [4].

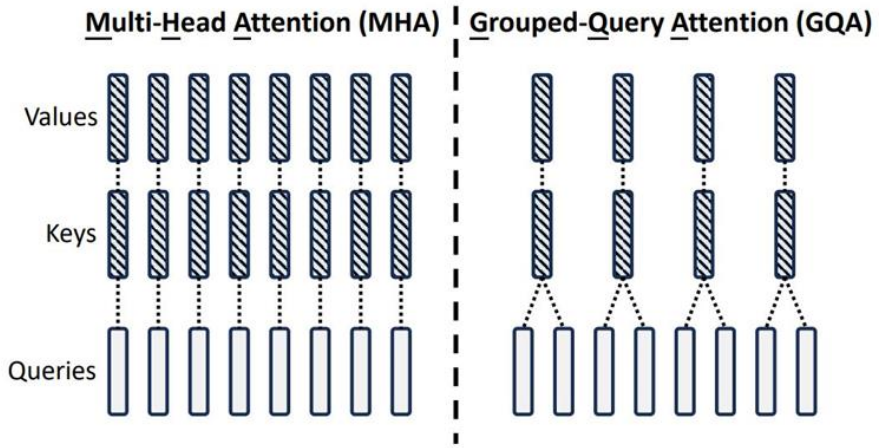
$$\text{head}_1 = \text{Attention}\left(\mathbf{Q}\mathbf{W}_1^Q, \mathbf{K}\mathbf{W}_1^K, \mathbf{V}\mathbf{W}_1^V\right)$$
$$\vdots$$
$$\text{head}_H = \text{Attention}\left(\mathbf{Q}\mathbf{W}_H^Q, \mathbf{K}\mathbf{W}_G^K, \mathbf{V}\mathbf{W}_G^V\right)$$

$$\text{MultiHeadAtt}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)$$

Inputs and outputs of each layer are the same dimensions:

$$\mathbf{Q} \in \mathbb{R}^{T \times d_{\text{model}}}$$
$$\mathbf{K} \in \mathbb{R}^{T/(\frac{H}{G}) \times d_{\text{model}}}$$
$$\mathbf{V} \in \mathbb{R}^{T/(\frac{H}{G}) \times d_{\text{model}}}$$

→ Reduce K, V cache storage to group sizes



Attention: Efficient Attention Mechanisms

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Multi-Query Attention: Single K, V for all Q heads [5]

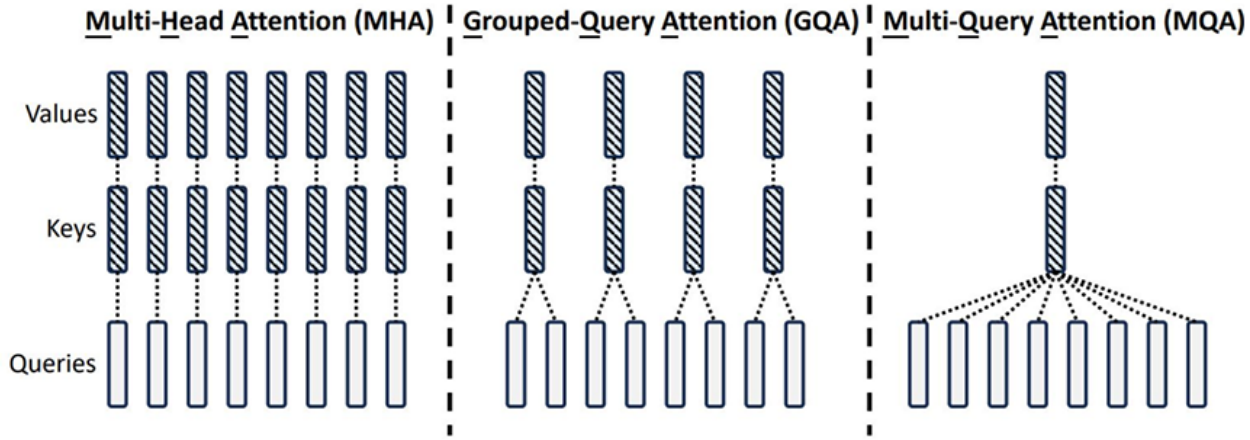
$$\text{head}_1 = \text{Attention}(\mathbf{Q}\mathbf{W}_1^Q, \mathbf{K}\mathbf{W}_1^K, \mathbf{V}\mathbf{W}_1^V)$$
$$\vdots$$
$$\text{head}_H = \text{Attention}(\mathbf{Q}\mathbf{W}_H^Q, \mathbf{K}\mathbf{W}_1^K, \mathbf{V}\mathbf{W}_1^V)$$

$$\text{MultiHeadAtt}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)$$

Inputs and outputs of each layer are the same dimensions:

$$\mathbf{Q} \in \mathbb{R}^{T \times d_{\text{model}}}$$
$$\mathbf{K} \in \mathbb{R}^{T/H \times d_{\text{model}}}$$
$$\mathbf{V} \in \mathbb{R}^{T/H \times d_{\text{model}}}$$

→ Further Reduce K, V Cache Size



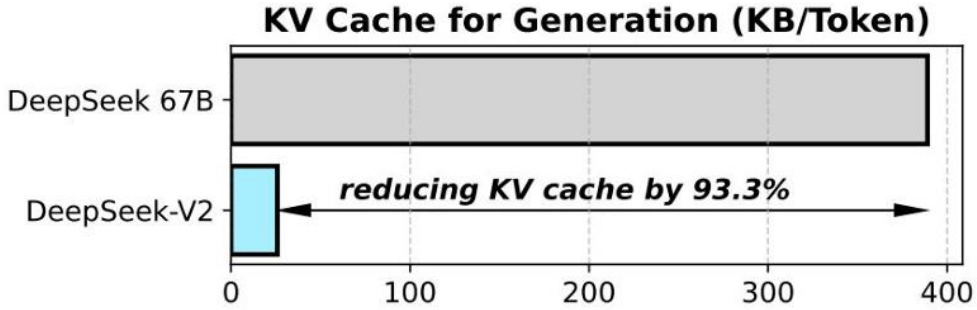
Attention: Efficient Attention Mechanisms

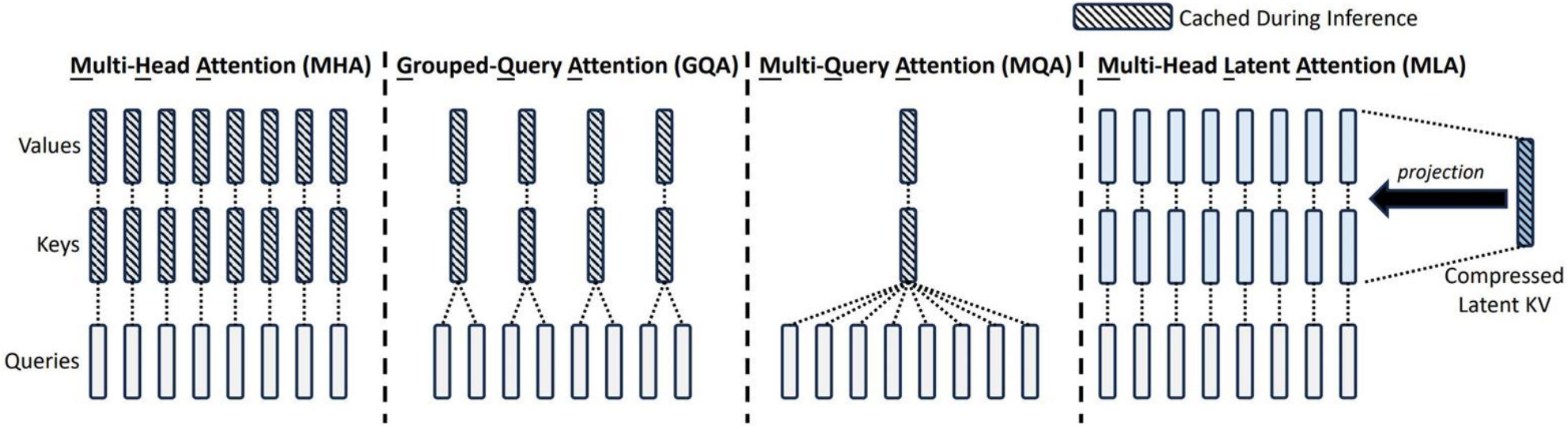
南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Multi-head Latent Attention: Project K, V into lower dimension latent vector [6]

$$k_t = \mathbf{W}^{UK} c_t^{KV}$$
$$v_t = \mathbf{W}^{UV} c_t^{KV}$$

$$c_t^{KV} = \mathbf{W}^{DKV} \mathbf{h}_t$$ Only latent vector to store

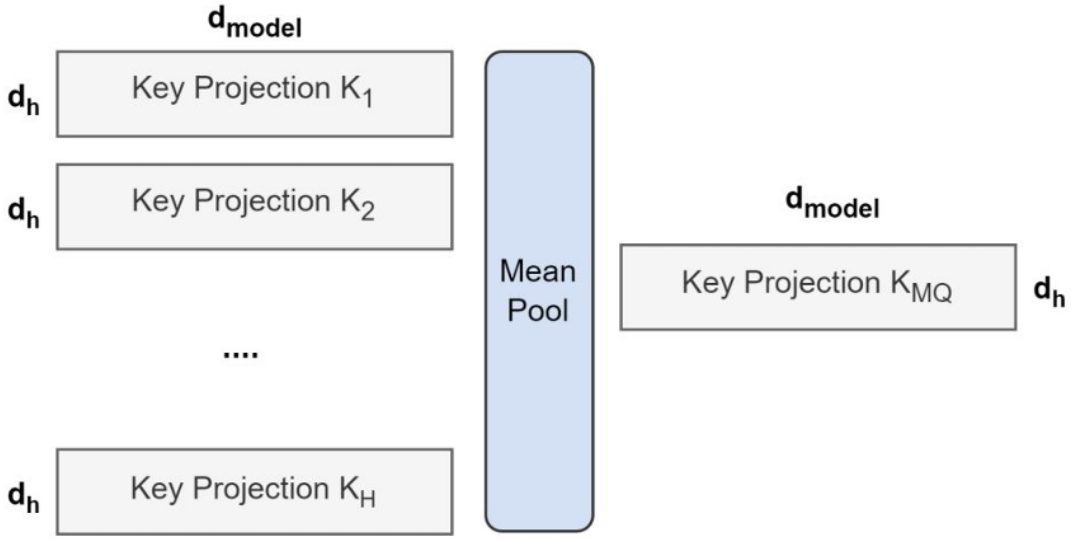




Attention: Efficient Attention Mechanisms

How to use efficient attention mechanisms:

- Pre-training directly with updated architecture
- Or use it as a compression method to speed up a rich multi-head attention model



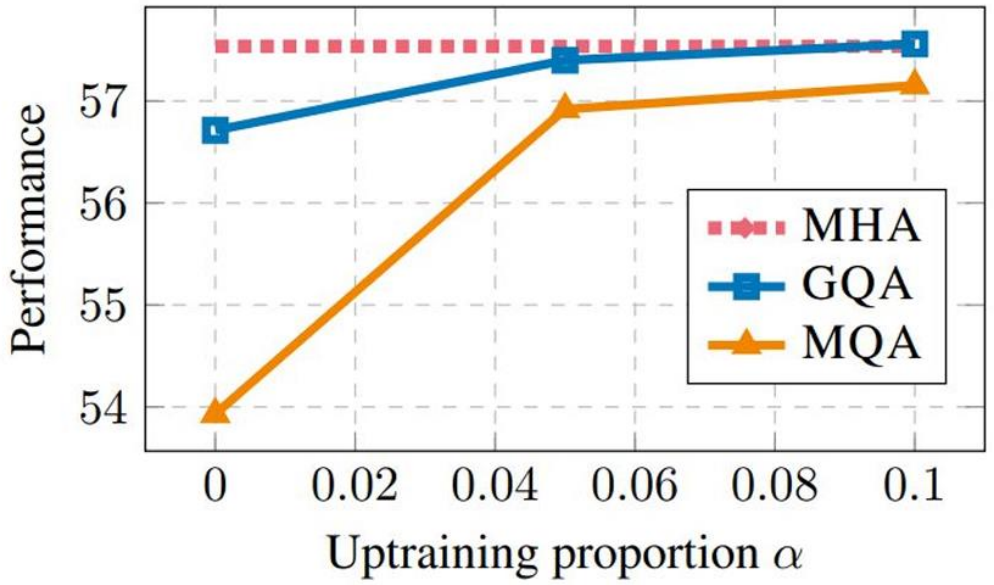
Mean Pooling Multi-Head Attention to Grouped-Query Attention [4]

Attention: Efficient Attention Mechanisms

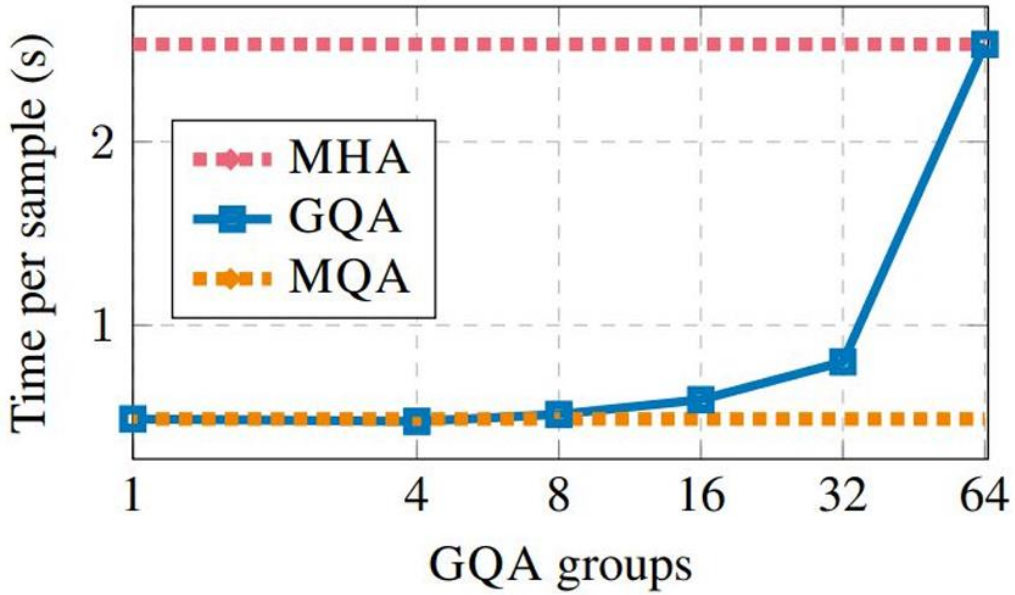
南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Performance:

- Recovering similar effectiveness as multi-head attention
- Significantly improve generation speed



Performance of Grouped-Query Attention Adapted from Multi-Head Attention [4]



Generation Efficiency with Grouped-Query Attention [4]

Attention: Sliding Window Attention (SWA)

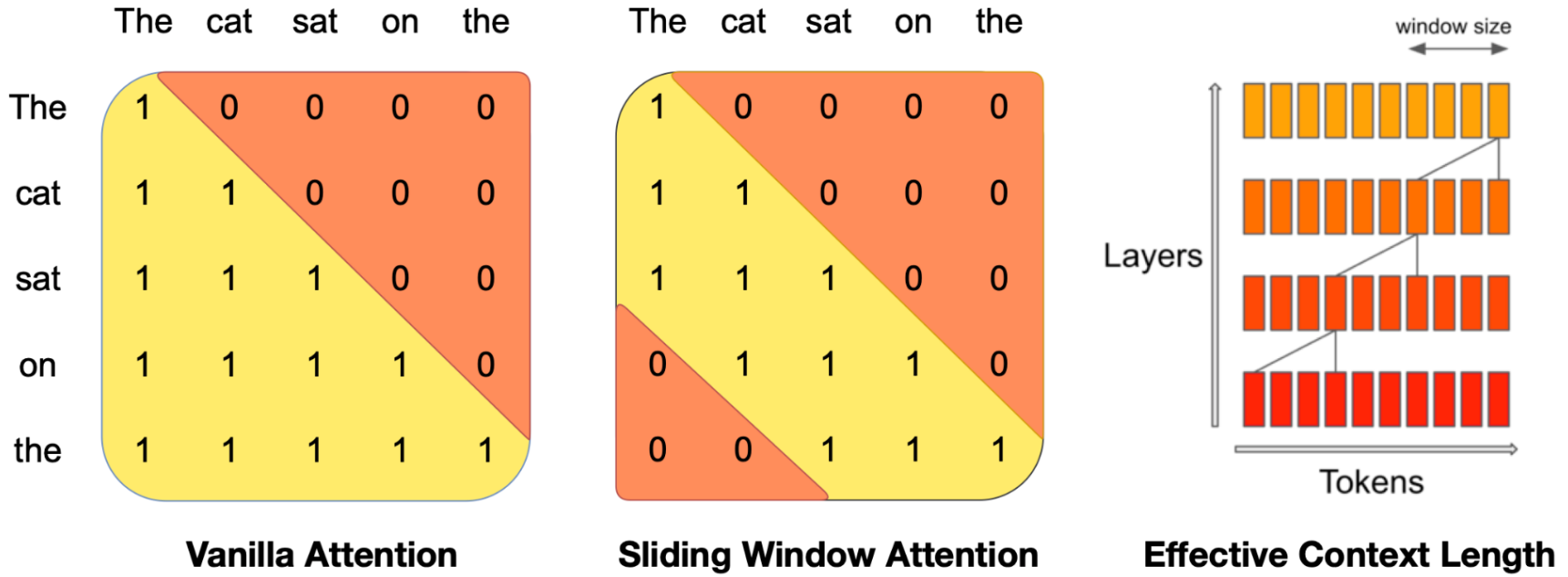
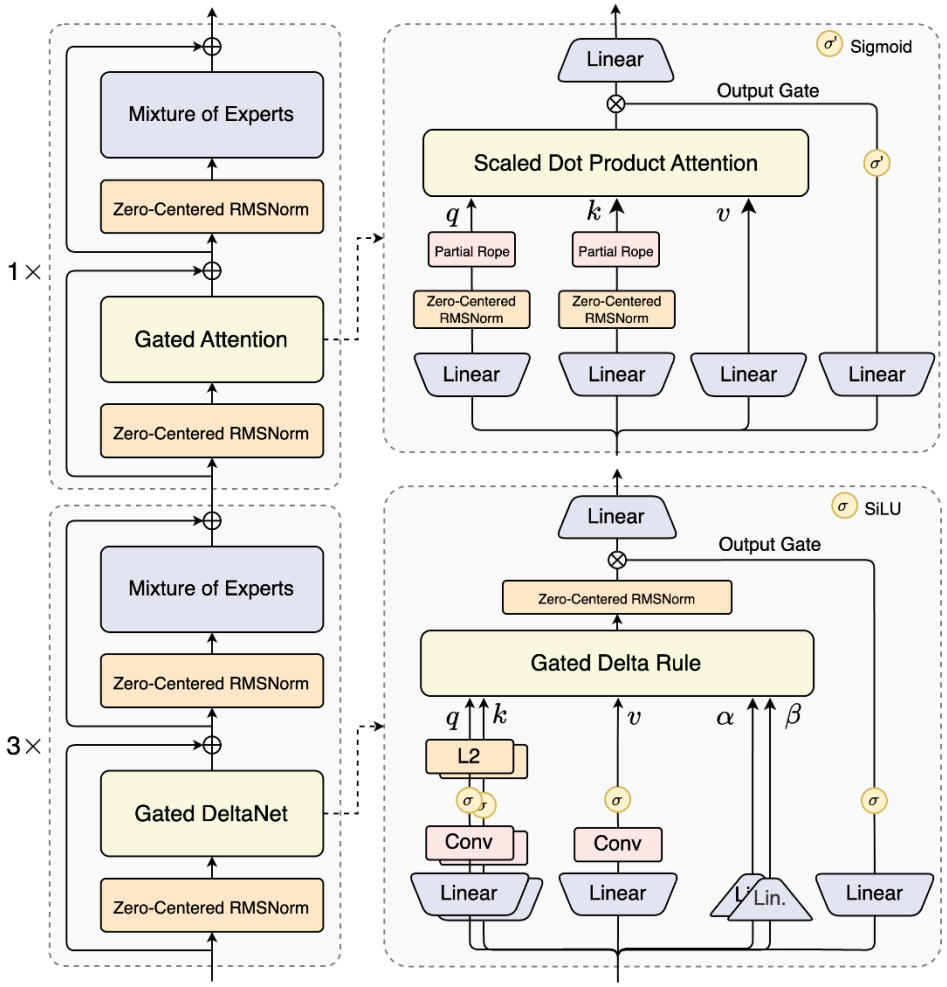


Figure 1: Sliding Window Attention. The number of operations in vanilla attention is quadratic in the sequence length, and the memory increases linearly with the number of tokens. At inference time, this incurs higher latency and smaller throughput due to reduced cache availability. To alleviate this issue, we use sliding window attention: each token can attend to at most W tokens from the previous layer (here, $W = 3$). Note that tokens outside the sliding window still influence next word prediction. At each attention layer, information can move forward by W tokens. Hence, after k attention layers, information can move forward by up to $k \times W$ tokens.

Mistral use GQA with SWA

Hybrid Architecture



Gated DeltaNet + Gated Attention (Qwen3-Next)

Pre-Norm and Post-Norm

- Post-LayerNorm
  - Used in original Transformer

Post-LN Transformer

$$
\begin{aligned}
x_{l,i}^{post,1} &= \text{MultiHeadAtt}(x_{l,i}^{post}, [x_{l,1}^{post}, \cdots, x_{l,n}^{post}]) \\
x_{l,i}^{post,2} &= x_{l,i}^{post} + x_{l,i}^{post,1} \\
x_{l,i}^{post,3} &= \text{LayerNorm}(x_{l,i}^{post,2}) \\
x_{l,i}^{post,4} &= \text{ReLU}(x_{l,i}^{post,3} W^{1,l} + b^{1,l}) W^{2,l} + b^{2,l} \\
x_{l,i}^{post,5} &= x_{l,i}^{post,3} + x_{l,i}^{post,4} \\
x_{l+1,i}^{post} &= \text{LayerNorm}(x_{l,i}^{post,5})
\end{aligned}
$$

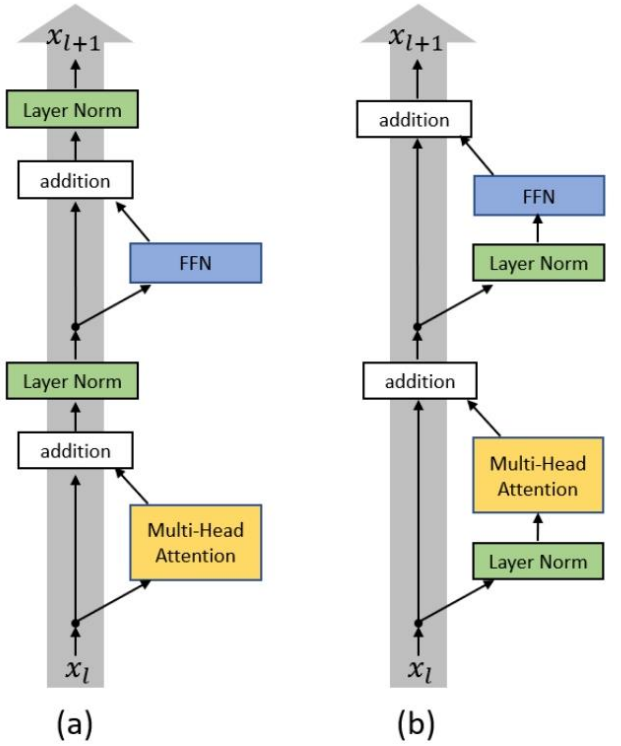
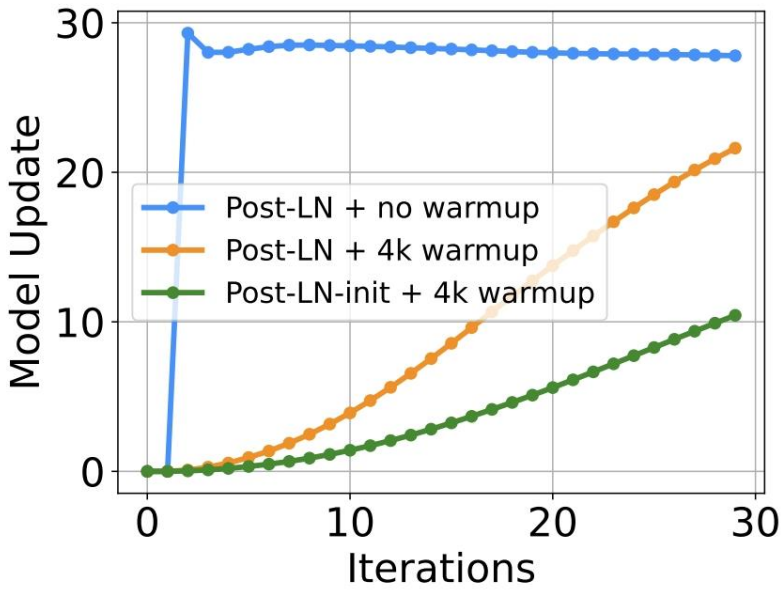


Figure 1. (a) Post-LN Transformer layer; (b) Pre-LN Transformer layer.

[2002.04745] On Layer Normalization in the Transformer Architecture

Pre-Norm and Post-Norm

- Post-LayerNorm
  - The expected gradients of the parameters near the output layer are large at the beginning of the optimization
  - Difficult to be optimized for deep layers
  - Need warmup and good parameter initialization
  - Better performance than Pre-LayerNorm



(a) Accumulated model update

Nearly no update shortly. The model has been stuck in a spurious local optima.

[2002.04745] On Layer Normalization in the Transformer Architecture

Pre-Norm and Post-Norm

- Pre-LayerNorm
  - Uses a final layer normalization right before the prediction
  - The gradients are well behaved without any exploding or vanishing at initialization
  - Can remove warmup period and more stable for deeper layers

Pre-LN Transformer

$$
\begin{aligned}
x_{l,i}^{pre,1} &= \text{LayerNorm}(x_{l,i}^{pre}) \\
x_{l,i}^{pre,2} &= \text{MultiHeadAtt}(x_{l,i}^{pre,1}, [x_{l,1}^{pre,1}, \cdots, x_{l,n}^{pre,1}]) \\
x_{l,i}^{pre,3} &= x_{l,i}^{pre} + x_{l,i}^{pre,2} \\
x_{l,i}^{pre,4} &= \text{LayerNorm}(x_{l,i}^{pre,3}) \\
x_{l,i}^{pre,5} &= \text{ReLU}(x_{l,i}^{pre,4} W^{1,l} + b^{1,l}) W^{2,l} + b^{2,l} \\
x_{l+1,i}^{pre} &= x_{l,i}^{pre,5} + x_{l,i}^{pre,3}
\end{aligned}
$$

Final LayerNorm: $x_{Final,i}^{pre} \leftarrow \text{LayerNorm}(x_{L+1,i}^{pre})$

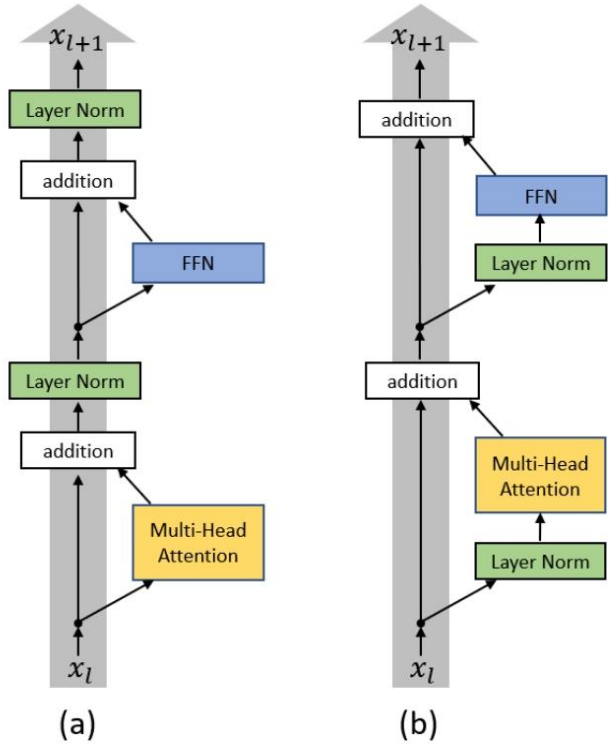
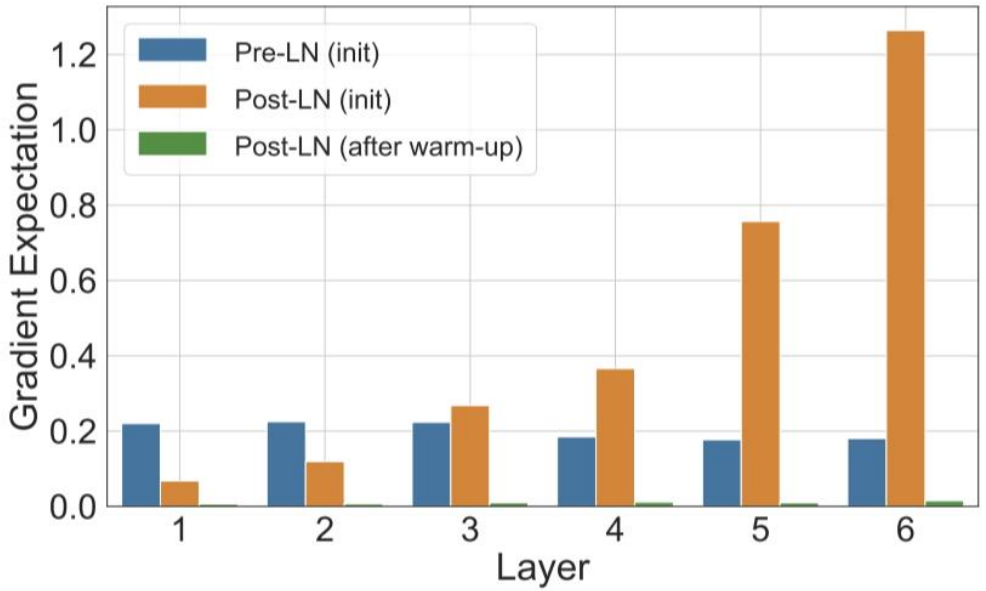


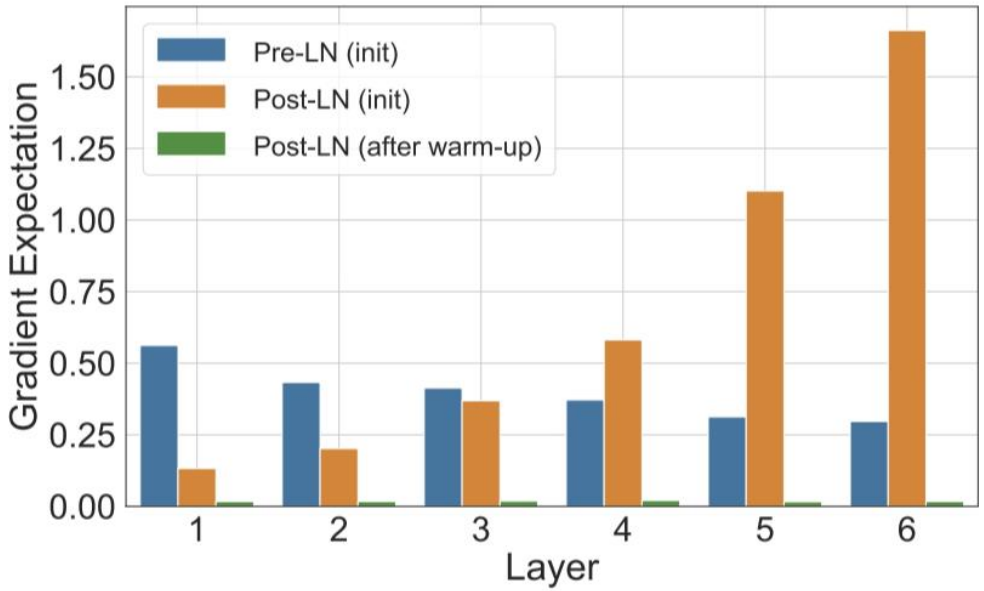
Figure 1. (a) Post-LN Transformer layer; (b) Pre-LN Transformer layer.

[2002.04745] On Layer Normalization in the Transformer Architecture

Pre-Norm and Post-Norm



(a) W^1 in the FFN sub-layers



(b) W^2 in the FFN sub-layers

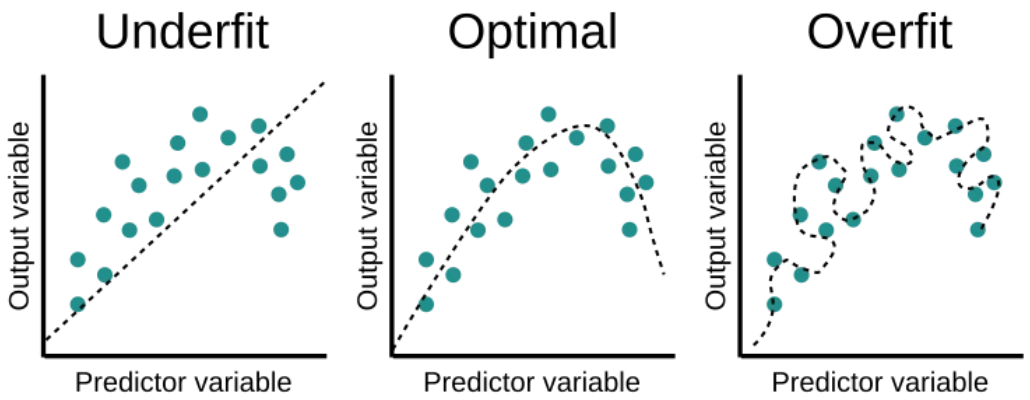
Figure 3: Norm of expected gradients for Pre-LN/Post-LN Transformer

Weight Decay

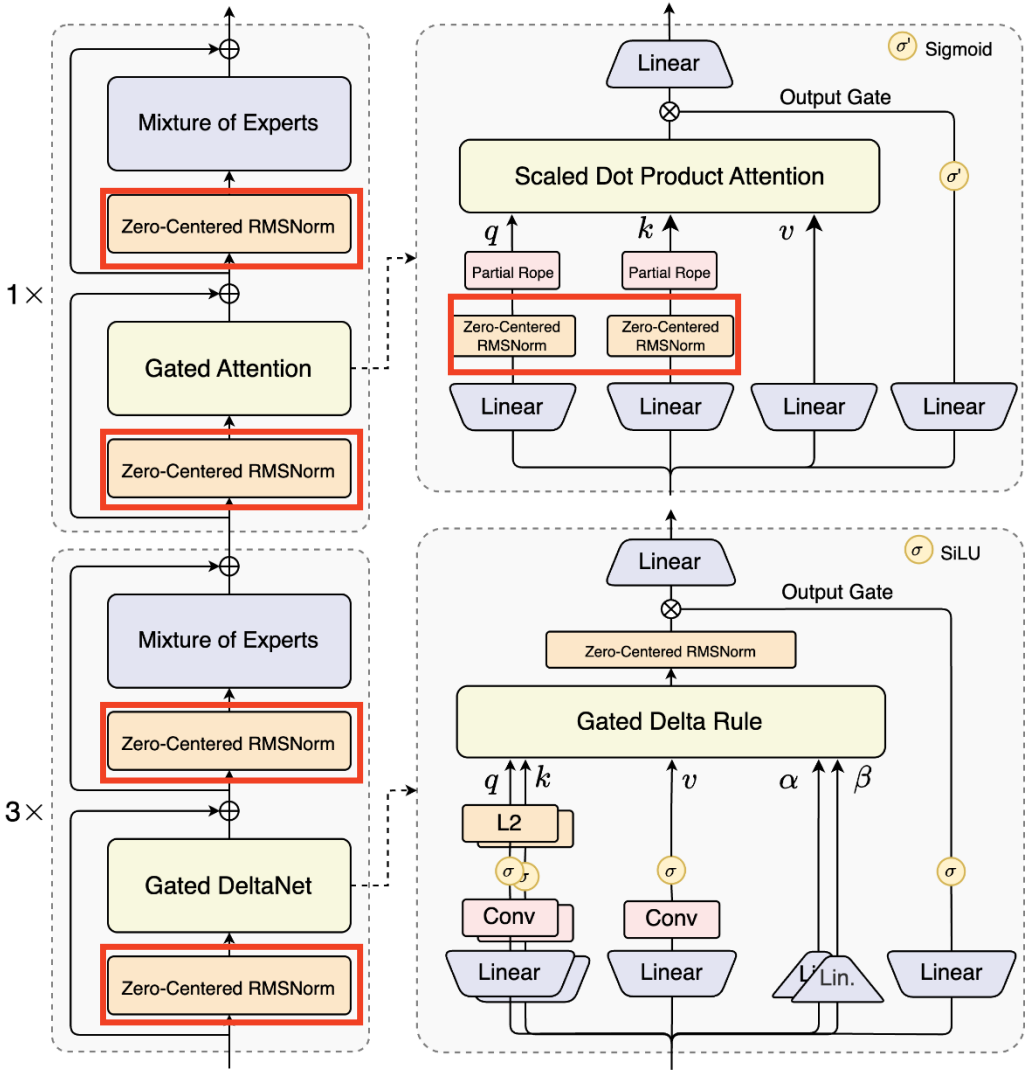
- Adding an L2 normalization to the loss function

$$\mathcal{L} = \mathcal{L} + \lambda\theta^T\theta$$

- Weight decay hyper-parameter $\lambda = 1e^{-4} \sim 1e^{-2}$
  - Too much weight decay might cause the model to underfit
  - The model may overfit when λ is too small
  - Usually set on a logarithmic scale
- Weight decay in vanilla SGD
  - $g_i^* = \frac{\partial \mathcal{L}}{\partial \theta_i} + 2\lambda\theta_i$
  - $\theta_{i+1} = \theta_i - \eta g_i^* = \left(\theta_i - \eta \frac{\partial \mathcal{L}}{\partial \theta_i}\right) - \eta \cdot 2\lambda\theta_i,$
- Weight decay parameter in Adam
  - Adam — PyTorch 2.1 documentation



Zero-Centered RMSNorm (Qwen3-Next)



- **Training-Stability-Friendly Designs**

We found that the attention output gating mechanism helps eliminate issues like Attention Sink [5] and Massive Activation [6] , ensuring numerical stability across the model.

In Qwen3, we use QK-Norm, but notice some layer norm weights become abnormally large. To fix this and further improve stability, Qwen3-Next adopts Zero-Centered RMSNorm [7], and applies weight decay to norm weights to prevent unbounded growth.

We also normalize MoE router parameters during initialization [8] , ensuring each expert is unbiasedly selected early in training — reducing noise from random initialization.

These stability-focused designs make small-scale experiments more reliable and help large-scale training run smoothly.

Alibi Position Embedding

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Attention with Linear Biases (ALiBi)
  - Relative position method
- Does not add positional embeddings to word embeddings
- It biases query-key attention scores with a penalty that is proportional to their distance
- After the query-key dot product, add a static, non-learned bias

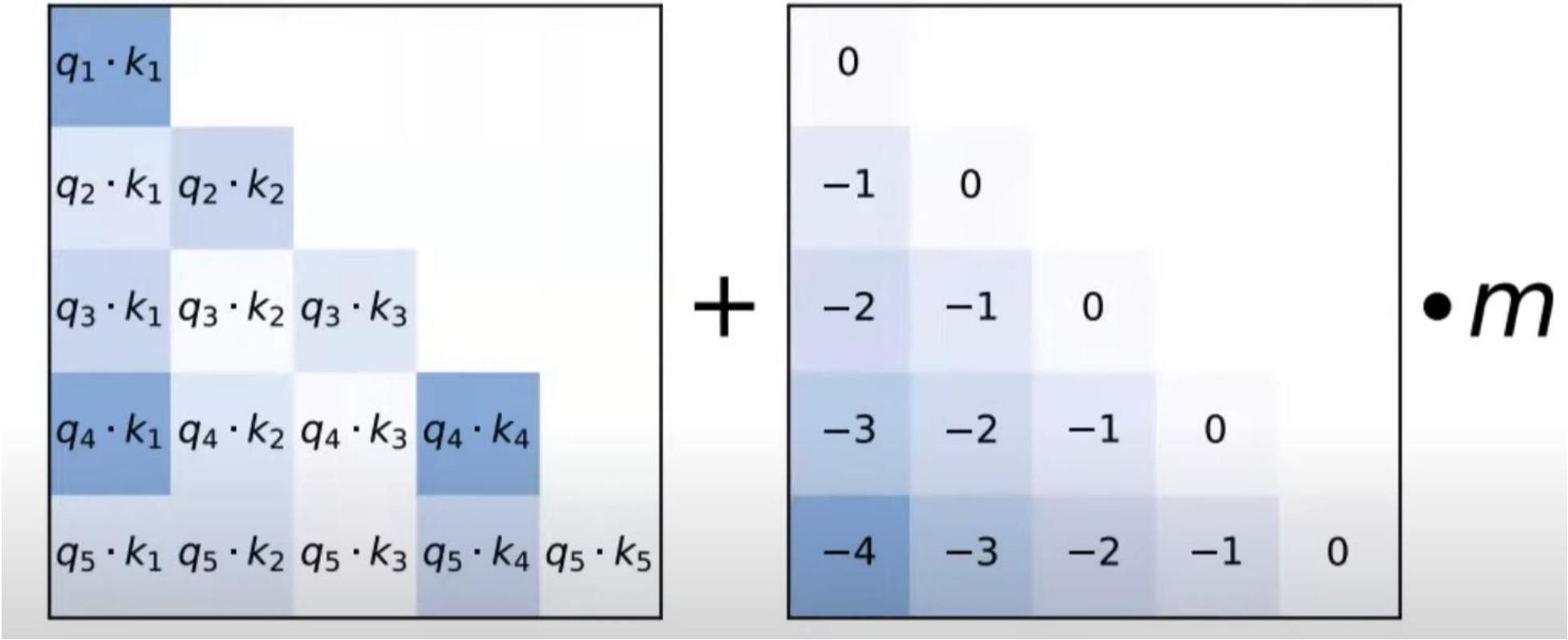
$$\text{softmax}(\mathbf{q}_i\mathbf{K}^\top + m \cdot [-(i-1), ..., -2, -1, 0]),$$

where scalar m is a head-specific slope fixed before training. Figure 3 offers a visualization.

[2108.12409] Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation (arxiv.org)

Alibi Position Embedding

- Implement it by modifying the mask matrix by adding the linear biases to it



[2108.12409] Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation (arxiv.org)

Alibi Position Embedding

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- For our models with 8 heads, the slopes are the geometric sequence $\frac{1}{2^1}, \frac{1}{2^2}, \ldots, \frac{1}{2^8}$
- For models that require n heads, interpolate those n slopes by geometrically averaging every consecutive pair $\frac{1}{2^{\frac{8}{n}}}, \frac{1}{2^{\frac{2\times 8}{n}}}, \ldots$
- It penalizes attention scores between distant query-key pairs, with the penalty increasing as the distance between a key and a query grows
- The different heads increase their penalties at different rates, depending on the slope magnitude.

ofirpress/attention_with_linear_biases: Code for the ALiBi method for transformer language models (ICLR 2022) (github.com)

[2108.12409] Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation (arxiv.org)

RoPE

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Rotary Position Embedding (RoPE)

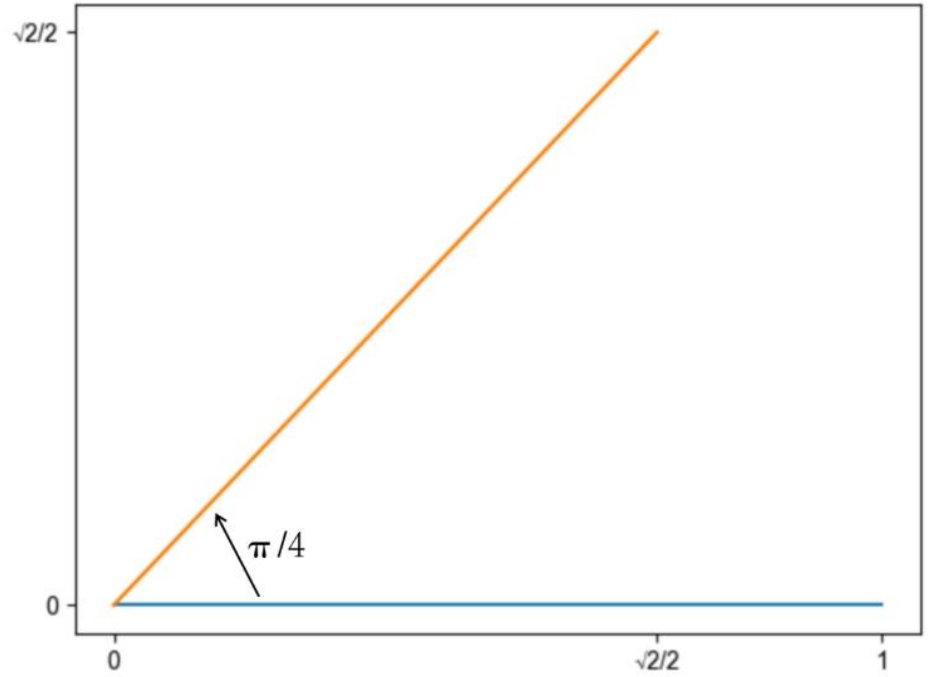
- Position information is encoded by rotating the vector
- Relative position information is represented by the inner product of two vectors
  - Require the inner product of query q_m and key k_n to be formulated by a function g, which takes only the word embeddings x_m, x_n, and their relative position $m - n$ as input variables
  - How to find a function g that satisfies

$$\langle f_q(\boldsymbol{x}_m, m), f_k(\boldsymbol{x}_n, n) \rangle = g(\boldsymbol{x}_m, \boldsymbol{x}_n, m - n).$$

[2104.09864] RoFormer: Enhanced Transformer with Rotary Position Embedding (arxiv.org)

RoPE

$$\begin{pmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos\frac{\pi}{4} \\ \sin\frac{\pi}{4} \end{pmatrix} = \begin{pmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix}$$



$$f(q,m) = R_m q = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} q_0 \\ q_1 \end{pmatrix}$$

RoPE

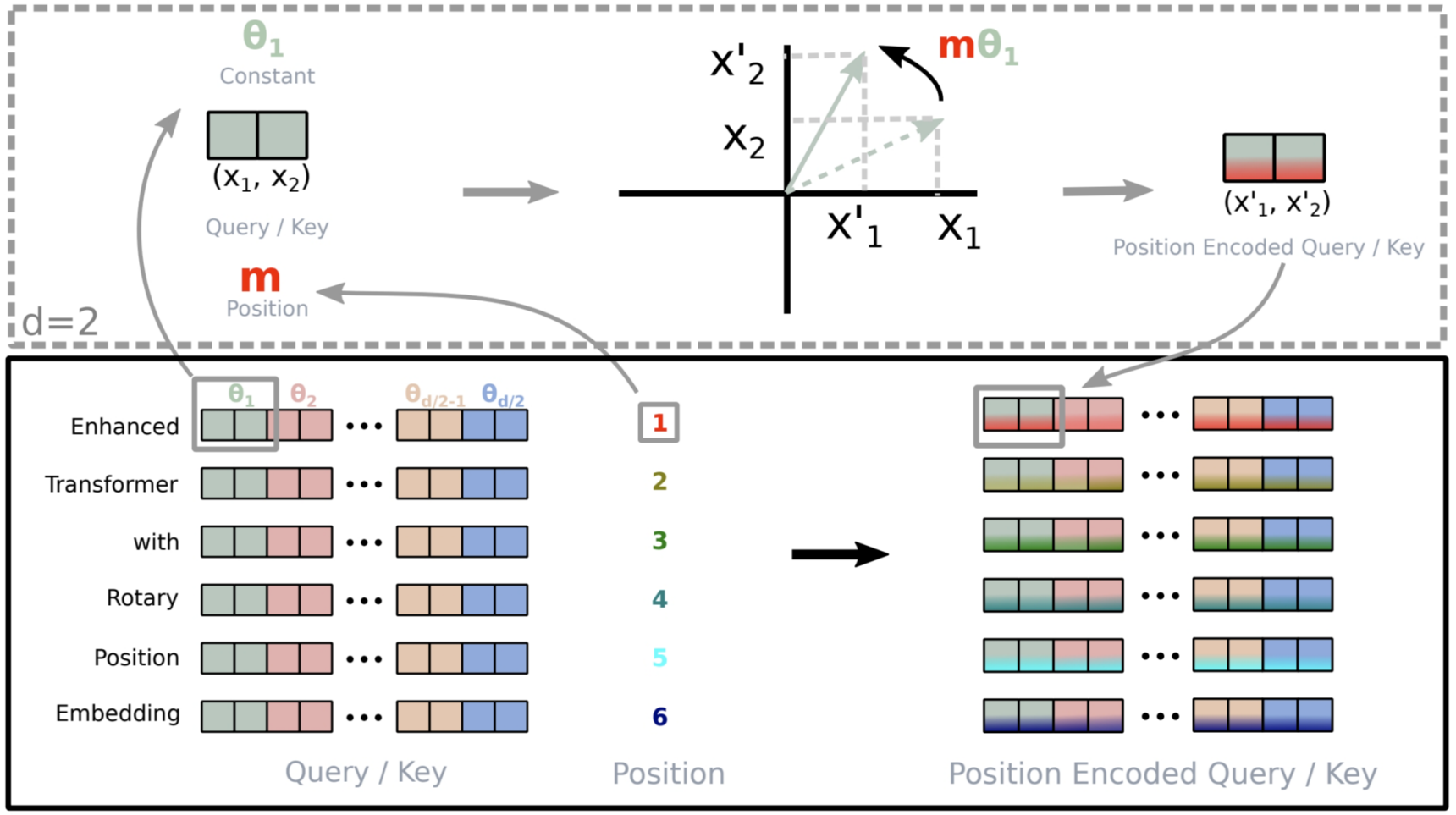


Figure 1: Implementation of Rotary Position Embedding(RoPE).

RoPE

$$\mathbf{x}_1, \cdots, \mathbf{x}_L \in \mathbb{R}^{|D|} \qquad f_{\{q,k\}}(\boldsymbol{x}_m, m) = \boldsymbol{R}^d_{\Theta,m} \boldsymbol{W}_{\{q,k\}} \boldsymbol{x}_m$$

$$\boldsymbol{R}^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

is the rotary matrix with pre-defined parameters $\Theta = \{\theta_i = 10000^{-2(i-1)/d}, i \in [1, 2, ..., d/2]\}$. A graphic illustration

$$\theta_d = b^{-2d/|D|} \text{ and } b = 10000$$

The attention weights decay as the relative distance m − n increases

transformers/src/transformers/models/roformer/modeling_roformer.py

RoPE

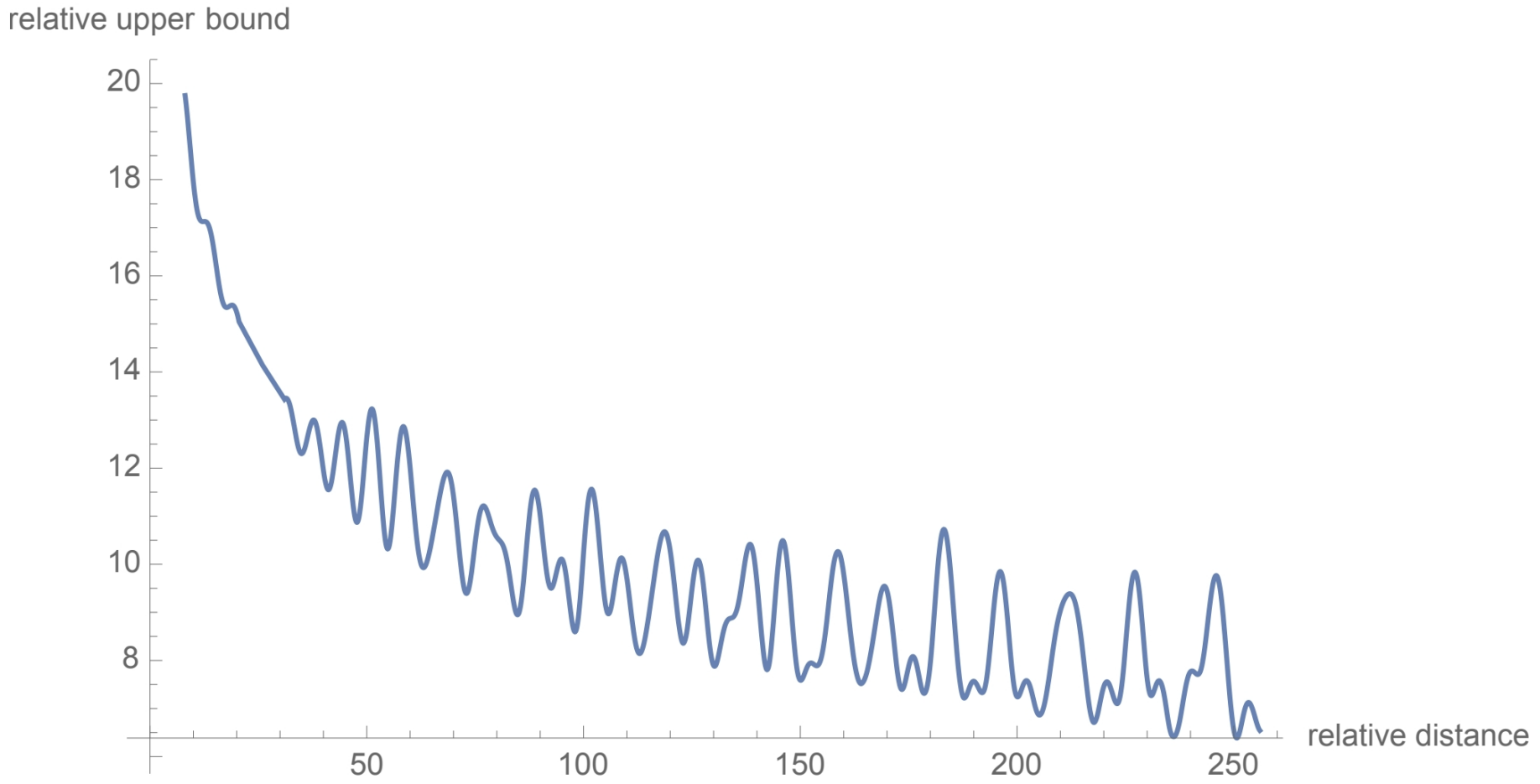


Figure 2: Long-term decay of RoPE.

Extend to Long Context

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Position Interpolation
  - Works well with the help of a small amount of fine-tuning

$$f_{\mathbf{W}}(\mathbf{x}_m, m, \theta_d) = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_l & -\sin m\theta_l \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_l & \cos m\theta_l \end{pmatrix} \mathbf{W}\mathbf{x}_m,$$

with the help of a small amount of fine-tuning. Specifically, given a pre-trained language model with RoPE, they modify the RoPE by

$$\theta_d = b^{-2d/|D|} \qquad f'_{\mathbf{W}}(\mathbf{x}_m, m, \theta_d) = f_{\mathbf{W}}\left(\mathbf{x}_m, \frac{mL}{L'}, \theta_d\right), \tag{10}$$

[2306.15595] Extending Context Window of Large Language Models via Positional Interpolation (arxiv.org)

Extend to Long Context

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

NTK-aware interpolation

- Define scale factor $s = \frac{L}{L'}$

$$f_{\mathbf{W}}(\mathbf{x}_m, m, \theta_d) = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_l & -\sin m\theta_l \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_l & \cos m\theta_l \end{pmatrix} \mathbf{W}\mathbf{x}_m,$$

$$g(m) = m$$

$$h(\theta_d) = b'^{-2d/|D|},$$

$$b' = b \cdot s^{\frac{|D|}{|D|-2}}.$$

$$f'_{\mathbf{W}}(\mathbf{x}_m, m, \theta_d) = f_{\mathbf{W}}(\mathbf{x}_m, g(m), h(\theta_d)),$$

NTK-Aware Scaled RoPE allows LLaMA models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation.

[2309.00071] YaRN: Efficient Context Window Extension of Large Language Models (arxiv.org)

Further Reading

- 浅谈Transformer的初始化、参数化与标准化 - 科学空间|Scientific Spaces
- Improving Transformer Optimization Through Better Initialization
- DeepNet: Scaling Transformers to 1,000 Layers [arxiv]

Thank you