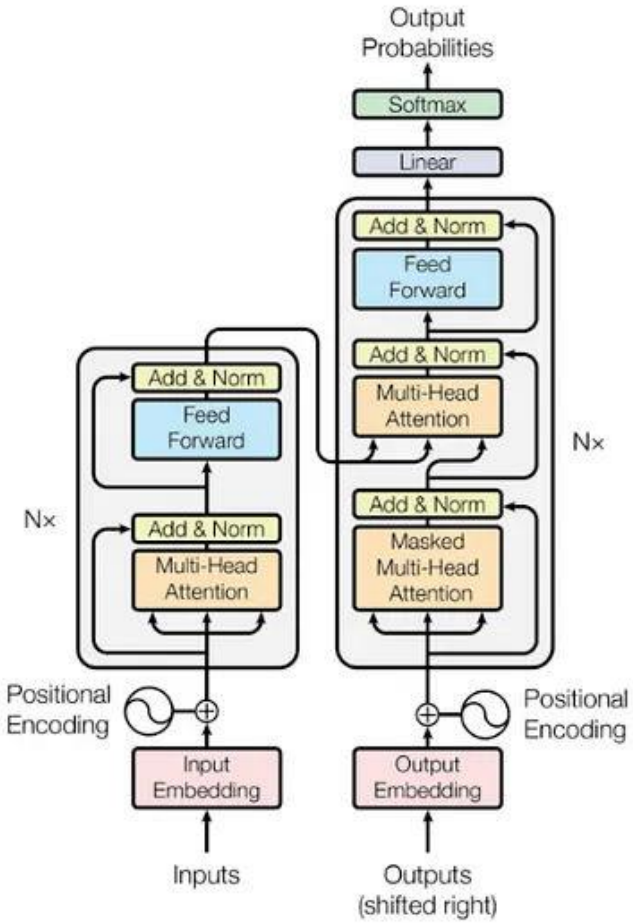
LLaMA模型结构解析

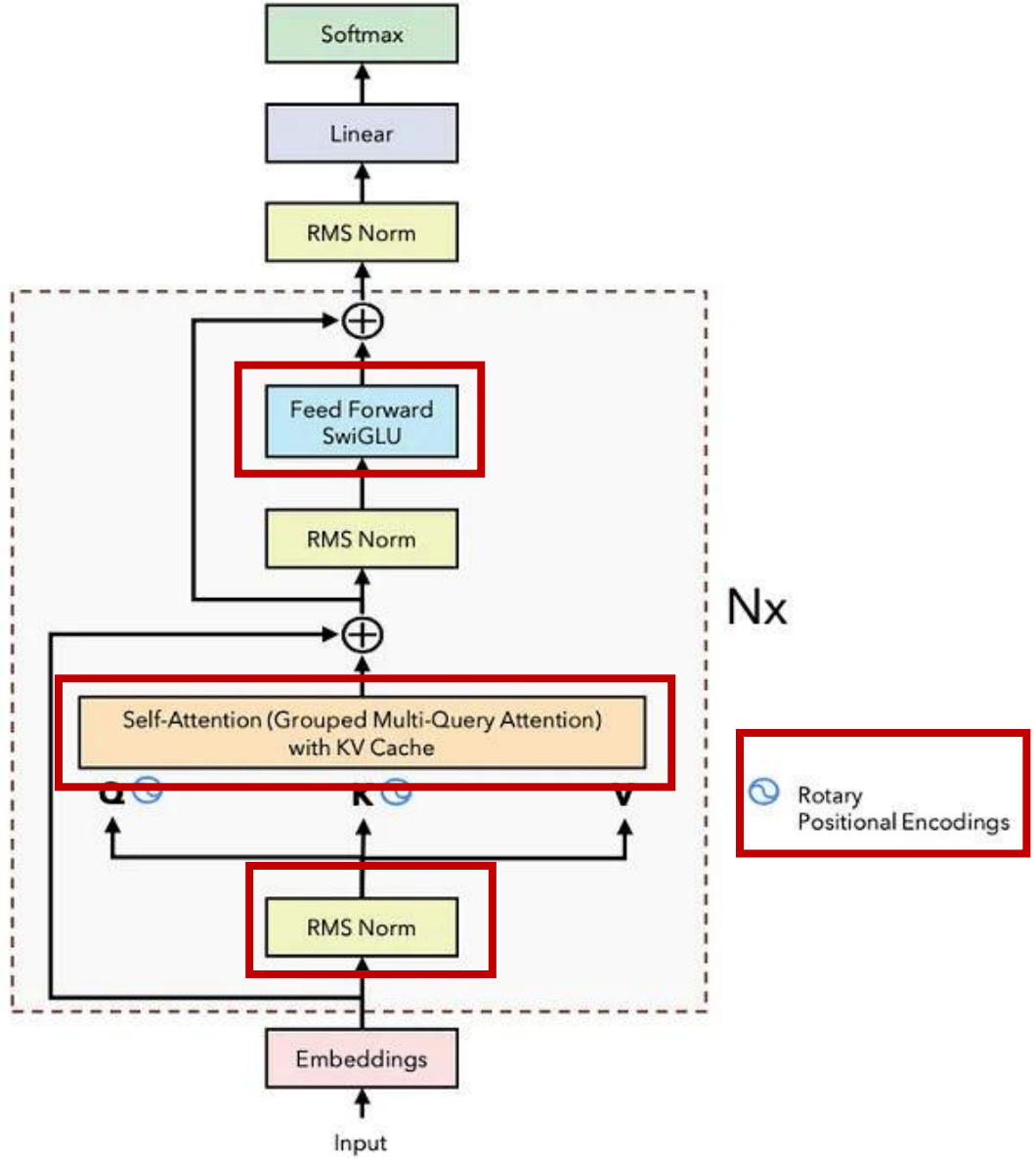
吴振一

郑州大学　河南先进技术研究院

Transformer vs LLaMA



Transformer
("Attention is all you need")



LLaMA

Outline

- RMSNorm
- SwiGLU
- RoPE
- Llama、Llama2、Llama3
- Group Attention

BatchNorm

对于输入特征 x ， Batch Normalization 的操作流程为：

1. **均值计算**：

$$\mu_{\text{batch}} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

2. **方差计算**：

$$\sigma^2_{\text{batch}} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\text{batch}})^2$$

3. **归一化**：

$$\hat{x}_i = \frac{x_i - \mu_{\text{batch}}}{\sqrt{\sigma^2_{\text{batch}} + \epsilon}}$$

4. **缩放和平移**：

$$y_i = \gamma \hat{x}_i + \beta$$

对一批次的向量进行归一化

LayerNorm

输入特征为 $x = [x_1, x_2, \ldots, x_n]$（长度为 n 的向量），LN 的计算过程如下：

1. **均值计算**：

$$\mu_{\text{layer}} = \frac{1}{n}\sum_{j=1}^{n} x_j$$

2. **方差计算**：

$$\sigma^2_{\text{layer}} = \frac{1}{n}\sum_{j=1}^{n}(x_j - \mu_{\text{layer}})^2$$

3. **归一化**：

$$\hat{x}_j = \frac{x_j - \mu_{\text{layer}}}{\sqrt{\sigma^2_{\text{layer}} + \epsilon}}$$

4. **缩放和平移**：

$$y_j = \gamma \hat{x}_j + \beta$$

对一个向量内的部分进行归一化

RMSNorm

1. **均方根计算**:

$$\mathrm{RMS}(x) = \sqrt{\frac{1}{n}\sum_{j=1}^{n} x_j^2}$$

2. **归一化**:

$$\hat{x}_j = \frac{x_j}{\mathrm{RMS}(x) + \epsilon}$$

3. **缩放**:

$$y_j = \gamma \hat{x}_j$$

RMSNorm 旨在用更简单的归一化方式，减少模型计算成本，同时在保证模型稳定性的前提下提高非线性表达能力

特性	Layer Normalization (LayerNorm)	Root Mean Square Normalization (RMSNorm)
归一化公式	$\hat{x} = \frac{x - \text{mean}(x)}{\sqrt{\text{var}(x) + \epsilon}}$	$\hat{x} = \frac{x}{\sqrt{\text{mean}(x^2) + \epsilon}}$
计算复杂度	相对较高，需要计算均值和方差	较低，只需计算均方根
依赖项	均值和方差计算依赖特征维度，适合低维向量	仅依赖于均方根，适合更高维度的输入
主要优点	能有效平衡不同特征维度的大小差异，提高模型收敛稳定性	计算量小，能在减少计算的同时达到类似效果
主要应用场景	适用于 RNN 和 Transformer 等对输入特征分布要求较高的场景	适合更大规模的模型，尤其在高维输入和加速训练时表现良好
效果对比	改进训练稳定性，尤其在 RNN 结构中	在高维度模型上效果与 LayerNorm 相似，且计算量较小

Outline

- RMSNorm
- SwiGLU
- RoPE
- Llama、Llama2、Llama3
- Group Attention

SwiGLU

- ReLU

  公式:

优点：简单的线性分段函数，不存在梯度消失问题，计算高效。
缺点：“神经元死亡”问题，对于负值输入，导数为零。

$$\mathrm{ReLU}(x) = \max(0, x)$$

- Swish

  公式:

$$\mathrm{Swish}(x) = x \cdot \sigma(x) = \frac{x}{1 + e^{-x}}$$

优点：提供更平滑的激活输出，并能自适应地将输入值缩放。
缺点：计算量稍大，由于其复杂性，模型收敛速度变慢。

SwiGLU

- GLU

  公式:

  优点：GLU 提供了灵活的门控机制，使得一部分输入能够控制另一部分的流通，从而提升模型的非线性表达能力。

$$\mathrm{GLU}(x) = x_1 \cdot \sigma(x_2)$$

其中 x_1 和 x_2 是输入向量的分块，x_1 用于生成主要输出，x_2 通过 Sigmoid 生成一个门控值。

- SwishGLU

  公式:

$$\mathrm{SwiGLU}(x) = x_1 \cdot \mathrm{Swish}(x_2) = x_1 \cdot (x_2 \cdot \sigma(x_2))$$

可以看作是将 GLU 的门控部分换成 Swish 函数，使得门控值更加平滑

SwiGLU

激活函数	公式	优点	缺点
ReLU	$\mathrm{ReLU}(x) = \max(0, x)$	计算简单，不存在梯度消失	导致“神经元死亡”问题
Swish	$\mathrm{Swish}(x) = x \cdot \sigma(x)$	平滑激活效果，减少信息丢失	计算量相对较大
GLU	$\mathrm{GLU}(x) = x_1 \cdot \sigma(x_2)$	增强信息控制能力	增加了计算量
SwiGLU	$\mathrm{SwiGLU}(x) = x_1 \cdot (x_2 \cdot \sigma(x_2))$	结合 Swish 和门控机制，增强控制力，计算高效	较复杂，适合大型模型

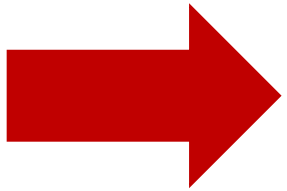
Outline

- RMSNorm
- SwiGLU
- RoPE
- Llama、Llama2、Llama3
- Group Attention

RoPE

公式： $\mathrm{RoPE}(x_k) = x_k \cdot e^{i\theta_k}$

- 位置 0：猫 → $\mathrm{PE}(0)$
- 位置 1：在 → $\mathrm{PE}(1)$
- 位置 2：沙发 → $\mathrm{PE}(2)$
- 位置 3：上 → $\mathrm{PE}(3)$
- 位置 4：睡觉 → $\mathrm{PE}(4)$



- 位置 0：猫 → $x_0 \cdot e^{i\theta_0}$
- 位置 1：在 → $x_1 \cdot e^{i(\theta_0+\Delta\theta)}$
- 位置 2：沙发 → $x_2 \cdot e^{i(\theta_0+2\Delta\theta)}$
- 位置 3：上 → $x_3 \cdot e^{i(\theta_0+3\Delta\theta)}$
- 位置 4：睡觉 → $x_4 \cdot e^{i(\theta_0+4\Delta\theta)}$

特性	普通位置嵌入（绝对）	旋转位置嵌入（相对）
位置关系建模方式	基于绝对位置索引	基于位置间的相对角度
泛化能力	对序列长度变化敏感	对长序列有更好适应性
公式	sin 和 cos 函数的绝对编码	旋转矩阵编码，使用复数或极坐标表示
应用场景	较适合固定长度序列	较适合长序列或动态序列

Outline

- RMSNorm
- SwiGLU
- RoPE
- Llama、Llama2、Llama3
- Group Attention

# params	7B	13B	33B	65B	7B	13B	34B	70B	8B			70B
# training tokens	1T	1T	1.4T	1.4T	2T	2T	2T	2T	15T			15T
hidden embed dim	4096	5120	6656	8192	4096	5120		8192	4096			8192
# attn heads	32	40	52	64	32	40		64	32			64
# attn layers	32	40	60	80	32	40		80	32			80
attention	MHA	MHA	MHA	MHA	MHA	MHA	GQA	GQA	GQA			GQA
# kv heads	32	40	52	64	32	40		8	8			8
mlp intermediate size	11008	13824	17920	22016	11008	13824		28672	14336			28672
context	2048				4096				8192			
tokenizer	BPE sentencepiece				BPE sentencepiece				BPE tiktoken			
token vocabulary	32000				32000				128256			
fine-tuned models	-				Llama-2-Chat (Jul 2023) Code Llama (Aug 2023)				Llama-3-Instruct (Apr 2024)			

BPE: Byte Pair Encoding

MHA: Multi-Head Attention

GQA: Grouped-Query Attention

Not released by Meta

Outline

- RMSNorm
- SwiGLU
- RoPE
- Llama、Llama2、Llama3
- Group Attention

1. **计算 Query, Key, Value**： 使用权重矩阵 W_Q, W_K, W_V 将 X 投影到 Query, Key, 和 Value 空间：

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

2. **计算 Attention Scores**： 通过 Query 和 Key 的内积计算注意力分数（在这里使用 Scaled Dot-Product Attention）：

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

其中 d_k 是 Key 的维度，用于缩放防止梯度爆炸。

1. **划分组**：将 X 分为 g 个子组：$X = [G_1, G_2, \ldots, G_g]$，其中每个 G_i 的大小为 m。
2. **组内 Attention 计算**： 对每个组 G_i 分别计算 Query, Key, Value：

$$Q_i = G_iW_Q, \quad K_i = G_iW_K, \quad V_i = G_iW_V$$

然后，分别对每个组计算注意力：

$$\text{Attention}(Q_i, K_i, V_i) = \text{softmax}\left(\frac{Q_iK_i^T}{\sqrt{d_k}}\right)V_i$$

3. **组合组的结果**：将每个组的注意力输出合并，得到最终的输出表示。

特性	传统 Attention	Group Attention
计算范围	整个序列	每组局部计算
计算复杂度	$O(n^2)$	$O(\frac{n^2}{g})$
适用场景	全局关系建模	局部关系建模，适合长序列
计算量	较大	较小，特别适合长序列和资源受限的环境