

大模型核心技术完全记忆手册

一、RoPE（旋转位置编码）

核心作用

- ****位置编码****: 为Transformer添加绝对+相对位置信息
- ****长度外推****: 训练后能处理更长序列
- ****数学特性****: 注意力分数仅与相对位置差有关

原理公式

向量分割: 将d维向量分成d/2组 $((x_0, x_1), (x_2, x_3) \dots)$ 旋转角度: $\theta_i = 10000^{-2i/d}$ 旋转矩阵: $R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ 最终计算: $q'_m = R(m\theta)q_m, k'_n = R(n\theta)k_n$

实现要点

```
```c
// 预计算cos/sin表
for (int i = 0; i < d/2; i++) {
 float theta = pow(10000.0, -2.0*i/d);
 cos_cache[pos][i] = cos(pos * theta);
 sin_cache[pos][i] = sin(pos * theta);
}

// 旋转计算 (每组两个元素)
float x0 = q[2*i], x1 = q[2*i+1];
q[2*i] = x0*cos - x1*sin; // 实部
q[2*i+1] = x0*sin + x1*cos; // 虚部
```

**记忆关键:** 向量旋转 → 保持模长 → 相对位置编码

---

## 二、Q/K投影 (ggml\_mul\_mat)

### 核心作用

- **空间映射**: 将输入映射到查询(Q)和键(K)空间
- **线性变换**:  $y = xW$ , 维度变换: hidden  $\rightarrow$  head\_dim $\times$ n\_heads

### 原理公式

```
Q = X × W_q [seq_len, hidden] × [hidden, head_dim×n_heads]
K = X × W_k [seq_len, hidden] × [hidden, head_dim×n_heads]
```

### 实现要点

```
// 矩阵乘法核心 (简化)
for (int i = 0; i < seq_len; i++) {
 for (int j = 0; j < out_dim; j++) {
 float sum = 0;
 for (int k = 0; k < in_dim; k++) {
 sum += x[i*in_dim + k] * w[j*in_dim + k];
 }
 out[i*out_dim + j] = sum;
 }
}
```

#### 优化技巧:

1. 权重分块加载, 减少缓存失效
2. 支持量化: INT8/INT4权重
3. 使用SIMD指令并行计算

---

### 三、RMS Norm (ggml\_rms\_norm)

#### 核心作用

- **稳定训练**: 替代LayerNorm, 加速收敛
- **简化计算**: 不减去均值, 仅用均方根
- **参数学习**:  $\gamma$ 缩放参数适应数据分布

#### 原理公式

```
RMS(x) = $\sqrt{\text{mean}(x^2) + \epsilon}$ # $\epsilon=1e-6$ 防止除零
x_norm = x / RMS(x)
输出 = $\gamma \odot x_norm$ # \odot 表示逐元素乘
```

#### 实现要点

```
// 三步计算流程
1. 计算平方和:
float ss = 0;
for (int i = 0; i < n; i++) ss += x[i] * x[i];

2. 计算缩放因子:
float scale = 1.0f / sqrtf(ss/n + 1e-6f);

3. 应用归一化和缩放:
for (int i = 0; i < n; i++)
 y[i] = x[i] * scale * gamma[i];
```

性能注意：一次遍历完成，向量化加速

## 四、Q/K Norm缩放（ggml\_mul）

### 核心作用

- **稳定softmax**：防止点积过大导致梯度消失
- **方差控制**：保持注意力分数方差≈1
- **数值稳定**：避免浮点数溢出

### 原理公式

注意力分数 = softmax( $Q \cdot K^T / \sqrt{d_k}$ )  
其中  $d_k$  = head\_dim (通常64/128)  
缩放因子 =  $1/\sqrt{d_k} \approx 0.125$

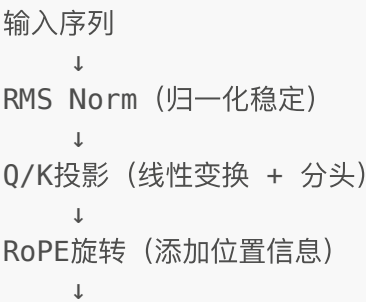
### 实现要点

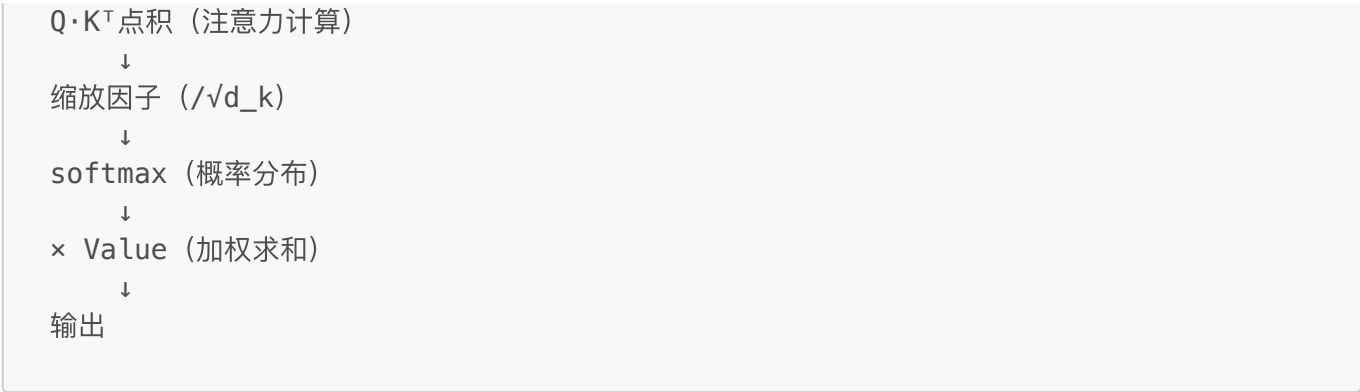
```
// 点积计算后立即缩放
float scale = 1.0f / sqrtf((float)head_dim);

for (int i = 0; i < seq_len; i++) {
 for (int j = 0; j < seq_len; j++) {
 // 计算点积
 float dot = 0;
 for (int k = 0; k < head_dim; k++) {
 dot += q[i*head_dim + k] * k[j*head_dim + k];
 }
 // 应用缩放
 attention[i][j] = dot * scale;
 }
}
```

记忆关键：点积后缩放 → 除 $\sqrt{d_k}$  → softmax稳定

## 完整计算流程





## 性能优化要点

### 1. 内存优化

- 数据连续存储 (行主序)
- 权重分块加载
- 量化压缩 (INT8/INT4)

### 2. 计算优化

- SIMD向量化并行
- 操作融合减少访存
- 预计算cos/sin表

### 3. 数值稳定

- RoPE: 高精度浮点避免累积误差
- Norm:  $\epsilon=1e-6$ 防止除零
- 缩放: softmax前必须缩放

### 4. 调试要点

- ✓ 检查旋转角度累加正确

✓ 验证RMS Norm无NaN/INF

✓ 确认缩放因子数值合理

✓ 监控量化误差<1%

## 速记口诀

### 核心流程:



### 技术要点:

位置编码用旋转，  
线性变换分头算。  
均方根化稳训练，  
缩放因子防溢出。  
量化分块加速跑，  
向量并行效率高。

### 数值关键：

$\theta$ 基：  $10000^{-2i/d}$   
 $\epsilon$ 值：  $1e-6$ 防除零  
缩放：  $1/\sqrt{d_k}$ 稳方差  
头维：  $64/128$ 是常见