典型大语言模型架构

清华大学计算机系 陈键飞

《大模型计算》课程团队

结构参数

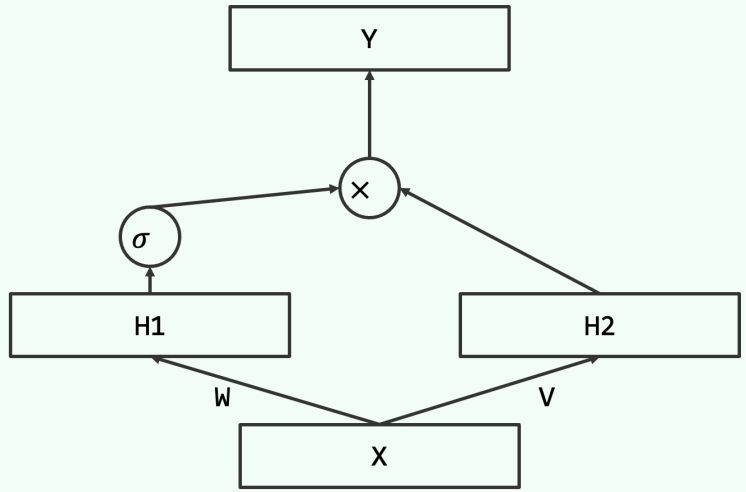
模型版本	参数量	层数	Hidden Dim	Heads	FFN Dim	Max Seq Len	Vocab Size
LLaMA-1	7B	32	4096	32	11008	2048	32000
LLaMA-1	13B	40	5120	40	13824	2048	32000
LLaMA-1	33B	60	6656	52	17920	2048	32000
LLaMA-1	65B	80	8192	64	22016	2048	32000
LLaMA-2	7B	32	4096	32	11008	4096	32000
LLaMA-2	13B	40	5120	40	13824	4096	32000
LLaMA-2	70B	80	8192	64	28672	4096	32000
LLaMA-3	1B	24	2048	16	5504	2048	128256
LLaMA-3	8B	32	4096	32	14336	8192	128256
LLaMA-3	70B	80	8192	64	28672	8192	128256
LLaMA-3	405B	128	12288	96	49152	128K	128256

FFN模块

```
class LlamaMLP(nn.Module):
    def __init__(self, config):
        …
        self.gate_proj = nn.Linear(self.hidden_size, self.intermediate_size, bias=config.mlp_bias)
        self.up_proj = nn.Linear(self.hidden_size, self.intermediate_size, bias=config.mlp_bias)
        self.down_proj = nn.Linear(self.intermediate_size, self.hidden_size, bias=config.mlp_bias)

    def forward(self, x):
        down_proj = self.down_proj(self.act_fn(self.gate_proj(x)) * self.up_proj(x))
        return down_proj
```

- 考虑Llama3-8B: D=4096, I=14336, L=32
- 参数量: 3 * H * I * L = 5.64B
- 计算量: 5.6B MACs = 11.3Gflops / token



https://github.com/huggingface/transformers/blob/7258ea44bc0c0a425a468f66f8559d1de8c4126d/src/transformers/models/llama/modeling_llama.py#L143

注意力模块

```
class LlamaAttention(nn.Module):
    def __init__(self, config: LlamaConfig, layer_idx: int):
        self.head_dim = getattr(config, "head_dim", config.hidden_size // config.num_attention_heads)
        self.num_key_value_groups = config.num_attention_heads // config.num_key_value_heads
        self.q_proj = nn.Linear(config.hidden_size, config.num_attention_heads * self.head_dim, bias=config.attention_bias)
        self.k_proj = nn.Linear(config.hidden_size, config.num_key_value_heads * self.head_dim, bias=config.attention_bias)
        self.v_proj = nn.Linear(config.hidden_size, config.num_key_value_heads * self.head_dim, bias=config.attention_bias)
        self.o_proj = nn.Linear(config.num_attention_heads * self.head_dim, config.hidden_size, bias=config.attention_bias)

    def forward(…):
        query_states = self.q_proj(hidden_states).view(hidden_shape).transpose(1, 2)
        key_states = self.k_proj(hidden_states).view(hidden_shape).transpose(1, 2)
        value_states = self.v_proj(hidden_states).view(hidden_shape).transpose(1, 2)
        attn_output, attn_weights = attention_interface(…)
        attn_output = attn_output.reshape(*input_shape, -1).contiguous()
        attn_output = self.o_proj(attn_output)
        return attn_output, attn_weights
```

https://github.com/huggingface/transformers/blob/7258ea44bc0c0a425a468f66f8559d1de8c4126d/src/transformers/models/llama/modeling_llama.py#L197

注意力模块

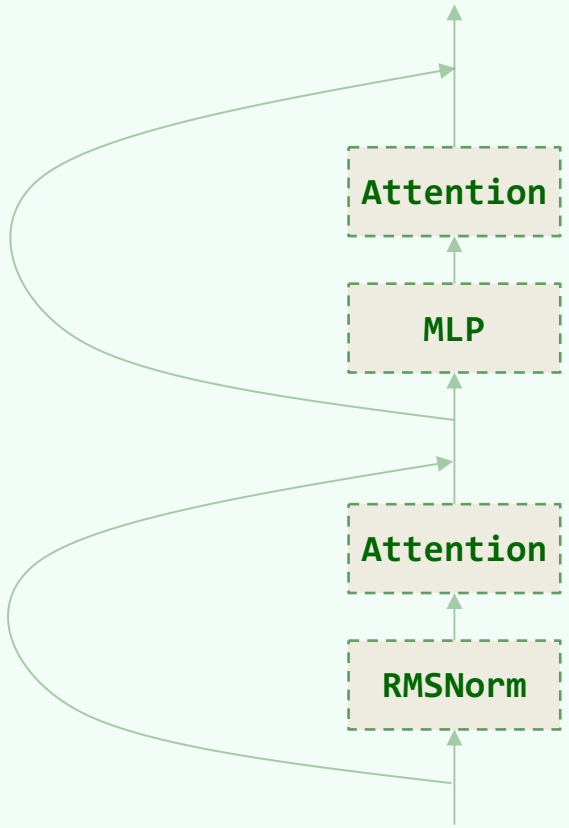
```
class LlamaAttention(nn.Module):
    def __init__(self, config: LlamaConfig, layer_idx: int):
        self.head_dim = getattr(config, "head_dim", config.hidden_size // config.num_attention_heads)
        self.num_key_value_groups = config.num_attention_heads // config.num_key_value_heads
        self.q_proj = nn.Linear(config.hidden_size, config.num_attention_heads * self.head_dim, bias=config.attention_bias)
        self.k_proj = nn.Linear(config.hidden_size, config.num_key_value_heads * self.head_dim, bias=config.attention_bias)
        self.v_proj = nn.Linear(config.hidden_size, config.num_key_value_heads * self.head_dim, bias=config.attention_bias)
        self.o_proj = nn.Linear(config.num_attention_heads * self.head_dim, config.hidden_size, bias=config.attention_bias)
```

- **考虑Llama3-8B：hidden_size=4096，num_attention_heads=32，num_kv_heads=8，head_dim=128，L=32**
- **参数量：(2*4096*4096 (QO) + 2*4096*8*128 (KV)) * 32 = 1.34B**
- **计算量（QKVO）：1.34B Macs = 2.68Gflops / token**
- **计算量（SDPA）：32层 * 32头 * 8192序列长度 * 128维 * 2(QK+PV) = 2.15GMacs = 4.30Gflops / token**
- **KV缓存大小：32层 * 8头 * 128维 * 8192序列长度 * 2（K+V） * 2字节 = 1GB**

https://github.com/huggingface/transformers/blob/7258ea44bc0c0a425a468f66f8559d1de8c4126d/src/transformers/models/llama/modeling_llama.py#L197

Transformer Block

```
class LlamaDecoderLayer(GradientCheckpointingLayer):
    def forward(…)
        residual = hidden_states
        hidden_states = self.input_layernorm(hidden_states)
        hidden_states, _ = self.self_attn(hidden_states, …)
        hidden_states = residual + hidden_states
        residual = hidden_states
        hidden_states = self.post_attention_layernorm(hidden_states)
        hidden_states = self.mlp(hidden_states)
        hidden_states = residual + hidden_states
        return hidden_states
```



https://github.com/huggingface/transformers/blob/7258ea44bc0c0a425a468f66f8559d1de8c4126d/src/transformers/models/llama/modeling_llama.py#L268

Transformer

```
class LlamaModel(LlamaPreTrainedModel):
  def forward(self, input_ids, …):
    hidden_states : torch.Tensor = self.embed_tokens(input_ids)
    for decoder_layer in self.layers[: self.config.num_hidden_layers]:
      hidden_states = decoder_layer(hidden_states, …)
    hidden_states = self.norm(hidden_states)
    return …

class LlamaForCausalLM(LlamaPreTrainedModel, GenerationMixin):
  def forward(self, input_ids, …):
    outputs= self.model(…)
    hidden_states = outputs.last_hidden_state
    logits = self.lm_head(hidden_states[:, slice_indices, :]
    return …
```

- **参数量:**
- **128256 * 4096 * 2 = 1.05B**

GPU

- RTX 4090
- 显存容量：24GB
- 显存带宽：1008 GB/s
- FP16/BF16算力：165 Tflops

- 模型flops利用率（Model Flops Utilization，MFU）
- 对于矩阵乘法（GEMM）来说，约为90%
  - 实际输出算力约150 Tflops
- 对于端到端训练，通常优化良好的实现在50%左右

参数量 / 计算量

- 参数量：共计8.03B
  - 词嵌入+LM head：1.05B
  - 全连接层： 5.64B（MLP）+ 1.34B（QKVO） = 6.98B
- 计算量：共计19.28Gflops / token（简易算法：每个参数2个flop）
  - 全连接层：11.3G（MLP） + 2.68G （QKVO） = 14.98G / token（全连接层参数量2倍）
  - SDPA： 4.30Gflops / token
- 显存开销：模型大小：8B * 2bytes = 16GB；KV缓存：1GB，共计17GB
- 预填充吞吐量：165T * 0.5 MFU / 19.28G = 4280 Token / s　　# 8K长度约2秒
- 训练吞吐量：4280 / 3 = 1427 Token / s
- 训完15T数据需要的时间：333卡*年

访存量

- 读取相应单词嵌入：D维向量 —— 可以忽略
- 计算线性层：访问全部模型权重：6.98B * 2Byte = 13GB
- 计算SDPA：访问全部KV缓存：1GB
- 访问整个LM head：128256 * 4096 * 2 Byte = 0.98GB
- 共计15GB / token（简易算法：每个参数2字节）

- 理想情况下，如果打满带宽，解码速度为1008 / 15 = 67 token/s